TRABALHO DE GRADUAÇÃO

DESENVOLVIMENTO DE ALGORITMOS VIA VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO DE LOCAL E POSTERIOR CONTROLE DE POUSO DE UM QUADRIRROTOR COMERCIAL

Eduardo de Mendonça Mesquita

Brasília, Julho de 2015

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

DESENVOLVIMENTO DE ALGORITMOS VIA VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO DE LOCAL E POSTERIOR CONTROLE DE POUSO DE UM QUADRIRROTOR COMERCIAL

Eduardo de Mendonça Mesquita

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Prof. Carlos Humberto Llanos Quintero, UnB/ ENM _____
(Orientador)
Prof. Renato Coral Sampaio, UnB/ FGA _____
(Co-orientador)
Prof. Carla Silva Rocha Aguiar, UnB/FGA _____
(Examinador interno)
Prof. João Yoshiyuki Ishihara, UnB/ENE _____
(Examinador interno)

Brasília, 10 de Julho de 2015

FICHA CATALOGRÁFICA

EDUARDO, DE MENDONÇA MESQUITA

Desenvolvimento de algoritmos via visão computacional para identificação de local e posterior controle de pouso de um quadrirrotor comercial,

[Distrito Federal] 2015.

xv, 71p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2015). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1.Processamento de imagens	2.Controle
3.Quadrirrotor	4.Pouso automático
I. Mecatrônica/FT/UnB	II. Desenvolvimento de algoritmos via visão computacional para identificação de local e posterior controle de pouso de um quadrirrotor comercial

REFERÊNCIA BIBLIOGRÁFICA

MESQUITA, E. M., (2015). DESENVOLVIMENTO DE ALGORITMOS VIA VISÃO COMPUTACIONAL PARA IDENTIFICAÇÃO DE LOCAL E POSTERIOR CONTROLE DE POUSO DE UM QUADRIRROTOR COMERCIAL. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 02, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 71p.

CESSÃO DE DIREITOS

AUTOR: Eduardo de Mendonça Mesquita.

TÍTULO DO TRABALHO DE GRADUAÇÃO: Desenvolvimento de algoritmos via visão computacional para identificação de local e posterior controle de pouso de um quadrirrotor comercial.

GRAU: Engenheiro ANO: 2015

_____

Eduardo de Mendonça Mesquita
SHCN EQ 406/407 BLOCO A – ASA NORTE
Brasília – DF – Brasil.

AGRADECIMENTOS

Primeiramente agradeço a Deus por me capacitar e abençoar em todos os caminhos trilhados nesta jornada nada fácil.

Agradeço aos meus pais, Francisco e Edna, que sem eles eu não teria o apoio necessário para me manter firme e não teria a base necessária para ter pleno gosto dos estudos.

À minha amada noiva Jéssica que se fez ao meu lado todo esse tempo, vencendo a dificuldade da distância e me dando forças para superar os desafios encontrados.

À minha irmã Letícia que sempre torceu por mim e me apoiou no que pôde.

Aos companheiros de jornada Leonardo, Marlon e Rédytton, que toparam em sair de Goiânia para cursar Engenharia em Brasília, essa parceria sem dúvida foi bastante importante durante esses anos.

Aos meus tios, Lellis e Hélia, que me auxiliaram no que foi preciso para me manter em Brasília.

Ao professor e orientador Carlos que aceitou o desafio e me ajudou a obter o resultado que vemos neste relatório

A todos aqueles que ao longo do curso fizeram parte da história e que também fazem parte desta conquista.

Eduardo de Mendonça Mesquita.

RESUMO

Este projeto tem como tema principal criar algoritmos em visão computacional com o objetivo de identificar um local para pouso de um quadrirrotor, o *Parrot* AR.Drone. Através de simulação foram projetados controladores para ação de movimentação do quadrirrotor a fim de obter controle preciso e respostas rápidas da trajetória para o pouso. A tarefa de identificar um alvo através de imagens, utilizando técnicas de processamento de imagens, não é simples quando se depara com as inúmeras interferências que o ambiente impõe, como a iluminação. Dessa forma definiram-se dois ambientes de trabalho, com pouca iluminação (noite) e iluminação abundante (dia). Nesses dois ambientes foram utilizadas técnicas diferentes de processamento de imagens, no ambiente escuro utilizou-se a segmentação por cor, já no ambiente claro foram utilizados algoritmos de extração e descrição de características, neste caso foi possível comparar dois métodos, o SURF (Speed-Up Robust Features) e o KLT (Kanade-Lucas-Tomasi Tracker). No projeto de controladores foi analisado dois movimentos do quadrirrotor, o *throttle* (subida e descida) e o *pitch* (arfagem ou guinada). Os controladores projetados foram por alocação de polos e zeros através do LGR (Lugar Geométrico das Raízes) e com resposta DeadBeat, com especificações de tempo e sobressinal máximo a serem alcançados. Todo o projeto foi definido então como um sistema realimentado em malha fechada.

Palavras Chave: Quadrirrotor, AR.Drone, pouso, processamento de imagens, controle.

ABSTRACT

This project's main theme creates algorithms in computer vision in order to identify a site for landing a quadrotor, the *Parrot* AR.Drone. Through simulation, controllers are designed for quadrotor move action in order to get precise control and quick responses of the trajectory for landing. The task of identifying a target through images, using image processing techniques, it is not easy when faced with the many interferences that the environment places, such as lighting. Thus it was established two working environments, low light (night) and abundant lighting (day). In these two environments were used different techniques of image processing, in the dark environment used the segmentation by color, as in clear environment were used algorithms for extraction and description of features, in this case it was possible to compare two methods, the SURF (Speed- up Robust Features) and KLT (Kanade-Lucas-Tomasi Tracker). In the control design was analyzed two movements of quadrirrotor, the throttle (up and down) and the pitch (forward motion). The controllers were designed for allocation of poles and zeros through the LGR (Locus Roots) and DeadBeat response, with specifications of maximum time and overshoot to be achieved. The entire project was then defined as a closed loop feedback system.

Keywords: Quadrotor; AR.Drone; landing field; image processing; control.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

A_P	Coeficiente de velocidade da hélice	[rad/s]
b	Coeficiente de empuxo	[Ns²]
B_P	Coeficiente da tensão de entrada do motor	[rad²/s²V]
C_P	Coeficiente constante do motor	[rad²/s²]
d	Coeficiente de arrasto	[Nms²]
e	Força eletromotriz do motor	[V]
$\boldsymbol{F^B}$	Vetor forças no quadrirrotor no sistema B	[N]
F_x	Força no eixo X	[N]
F_y	Força no eixo Y	[N]
F_z	Força no eixo Z	[N]
g	Aceleração da gravidade	[m/s²]
$\mathcal{H}$	Matriz Hessiana	
i	Corrente	[A]
I	Momento de inércia	[Nms²]
J_P	Momento de inércia da hélice	[Nms²]
J_{TM}	Momento de inércia total do motor	[Nms²]
J_{TP}	Momento de inércia total no eixo da hélice	[Nms²]
J_{TS}	Inércia total do sistema	[Nms²]
K_E	Constante do motor	[Vs/rad]
l	Distância do centro do quadrirrotor e o centro dos motores	[m]
L	Indutância	[H]
Lxx	Derivada parcial na horizontal da imagem	
Lxy	Derivada parcial na diagonal da imagem	
Lyy	Derivada parcial na vertical da imagem	
m	Massa do corpo	[kg]
M_p	Sobressinal do sinal da planta	
p	Velocidade angular no eixo X	[rad/s]
$\dot{p}$	Aceleração angular no eixo X	[rad/s²]
q	Velocidade angular no eixo Y	[rad/s]
$\dot{q}$	Aceleração angular no eixo Y	[rad/s²]

r	Velocidade angular no eixo Z	[rad/s]
$\dot{r}$	Aceleração angular no eixo Z	[rad/s²]
R	Resistência	[Ω]
T_L	Torque da carga	[Nm]
T_M	Torque do motor	[Nm]
T_{M_P}	Torque do motor aplicado ao eixo da hélice	[Nm]
T_P	Torque da hélice	[Nm]
T_{P_M}	Torque da hélice aplicado ao eixo do motor	[Nm]
t_s	Tempo de assentamento da resposta	
u	Velocidade linear no eixo X	[m/s]
$\dot{u}$	Aceleração linear no eixo X	[m/s²]
U	Força aplicada pelo motor	[N]
v	Velocidade linear no eixo Y	[m/s]
$\dot{v}$	Aceleração linear no eixo Y	[m/s²]
$\boldsymbol{v}$	Vetor de velocidade do quadrirrotor no sistema E	
v_L	Tensão sobre o indutor	[V]
v_R	Tensão sobre o resistor	[V]
v_t	Tensão total do circuito	[V]
$\boldsymbol{V^B}$	Vetor velocidade linear do quadrirrotor no sistema B	[m/s]
$\dot{\boldsymbol{V}}^{\boldsymbol{B}}$	Vetor aceleração linear do quadrirrotor no sistema B	[m/s²]
X	Posição no eixo X	[m]
Y	Posição no eixo Y	[m]
w	Velocidade linear no eixo Z	[m/s]
$\dot{w}$	Aceleração linear no eixo Z	[m/s²]
w_n	Frequência natural de oscilação do sistema	[rad/s]
Z	Posição no eixo Z	[m]
Z_a	Função de transferência para altitude	
Z_p	Função de transferência do movimento *pitch*	

Símbolos Gregos

$\boldsymbol{\zeta}$	Vetor de posição do quadrirrotor no sistema B	
$\dot{\boldsymbol{\zeta}}$	Vetor de velocidade do quadrirrotor no sistema B	
$\Gamma^{\mathbf{E}}$	Vetor de posição linear do quadrirrotor no sistema E	[m]

$\boldsymbol{\Theta^E}$	Vetor de posição angular do quadrirrotor no sistema E	[rad]
φ	Ângulo do quadrirrotor no eixo X	[rad]
θ_c	Ângulo a compensar pelo controlador	
θ	Ângulo do quadrirrotor no eixo Y	[rad]
ψ	Ângulo do quadrirrotor no eixo Z	[rad]
$\boldsymbol{\Lambda}$	Vetor total de forças no quadrirrotor	
Ω	Velocidade angular do motor	[rad/s]
$\boldsymbol{\Omega_P}$	Vetor de velocidade angular das hélices	[rad/s]
$\boldsymbol{\tau^B}$	Vetor de torque no quadrirrotor no sistema B	[Nm]
τ_x	Torque no eixo X	[Nm]
τ_y	Torque no eixo Y	[Nm]
τ_z	Torque no eixo Z	[Nm]
$\boldsymbol{\zeta}$	Vetor de posição do quadrirrotor	[m²/s]
ξ	Fator de amortecimento	
$\boldsymbol{\omega^B}$	Vetor velocidade angular do quadrirrotor no sistema B	[rad/s]
$\boldsymbol{\dot{\omega}^B}$	Vetor aceleração angular do quadrirrotor no sistema B	[rad/s²]

Grupos Adimensionais

J_θ	Matriz generalizada
K_M	Constante de proporcionalidade do motor
N	Razão de redução da engrenagem
η	Eficiência da energia do motor para a hélice
R_θ	Matriz de rotação
T_θ	Matriz de Translação

Sobrescritos

•	Primeira derivada temporal
B	Referência sistema B
E	Referência sistema E

Siglas

CES	(do inglês Consumer Eletronics Show)
CPU	(do inglês Central Processing Unit)

ECCV	(do inglês European Conference on Computer Vision)
KLT	(do inglês Kanade Lucas Tomasi Tracker)
MEMS	(do inglês Micro-Electro-Mechanical Systems)
ROI	(do inglês Region of Interest)
SIFT	(do inglês Scale Invariant Feature Transform)
SURF	(do inglês Speed-Up Robust Features)
VANT	Veículo Aéreo Não Tripulado

CAPÍTULO 1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Aeronaves são ferramentas indispensáveis na sociedade moderna, constituídos de diversos tipos de tecnologias embarcadas, estudados nas áreas de engenharia como elétrica e mecatrônica. Além daquelas aeronaves responsáveis pelo transporte de passageiros existem aeromodelos e os Veículos Aéreos Não Tripulados (VANT) controlados autonomamente via computador ou controle remoto [3]. Inicialmente os VANTS foram concebidos para o uso militar (Figura 1.1), atualmente os mesmos possibilitaram tarefas dificilmente realizadas pelo ser humano, como em caso de acesso a regiões avassaladas por tragédias ambientais ou por armamentos. [13]

Figura 1. 1 – Uso militar de VANTS[1].

Inicialmente construídos em asa fixa, os VANTS foram avançando até serem estabelecidos por quatro hélices, chamados por quadrirrotor. Essa configuração tornou o voo muito mais estável e de fácil manuseio. [3]

1.2 DEFINIÇÃO DO PROBLEMA

Sabemos que uma tarefa difícil de ser realizada é a automação de dispositivos cuja dinâmica por si só é complexa. Em um voo de um quadrirrotor diversas variáveis podem ser observadas e forte correlação entre elas. Na verdade, este tipo de comportamento complexo das variáveis é típico de sistemas não lineares.

[1] http://veja.abril.com.br/

Por esses motivos, fazer com que um quadrirrotor pouse de forma autônoma é um problema que envolve diversas áreas de pesquisa. Neste trabalho é apresentada uma proposta de solução para o problema.

1.3 OBJETIVOS

Este trabalho propõe o desenvolvimento de uma plataforma que simule a identificação de um local e o controle do pouso de um quadrirrotor. Todo o procedimento é dividido em duas etapas. A primeira aborda a identificação do local de pouso através das imagens obtidas pela câmera vertical do quadrirrotor. A segunda aborda o controle do movimento de arfagem e altitude a fim de estabelecer um deslocamento mais rápido até a região desejada.

Os objetivos específicos foram definidos dentro destas duas etapas, conforme itens abaixo:

1. Captura das imagens provenientes da câmera vertical do quadrirrotor;
2. análise e processamento das imagens para identificação do local de pouso;
3. transformação de coordenadas da imagem em coordenadas espaciais;
4. utilização de modelos identificados em outros trabalhos para síntese de controladores;
5. validação dos controladores projetados via simulação;
6. simulação de uma trajetória de pouso verificando a melhora no tempo total de ação.

A visão geral do sistema proposto pode ser visualizada pela Fig 1.2. A identificação do local de pouso passa coordenadas de localização para o controle do pouso do quadrirrotor.

Figura 1. 2 – Visão Geral do sistema proposto

O presente relatório tem por objetivo relatar todo o processo pelo qual o trabalho de graduação foi executado. Toda a teoria necessária para que os procedimentos sejam entendidos está aqui relatada, principalmente na área de processamento de imagens, onde os algoritmos são constituídos de etapas de análises, com uma determinada complexidade matemática e computacional.

Desta maneira as dificuldades existentes durante a elaboração do trabalho também estão aqui relatadas, bem como a forma encontrada de resolvê-las.

1.4 APRESENTAÇÃO DO MANUSCRITO

O capítulo 2 descreve as especificações do quadrirrotor *Parrot AR.Drone* utilizado neste trabalho, tais como, o comportamento dinâmico e toda a estrutura de hardware do mesmo. Além disso, é listado alguns dos programas, software e bibliotecas que foram utilizados a fim de obter o processamento de imagem, captura de vídeos diretamente do quadrirrotor e projeto dos controladores.

O capítulo 3 apresenta toda a modelagem matemática da dinâmica de um quadrirrotor, indicando os movimentos possíveis de serem realizados juntamente com as variáveis representativas. Também serão abordadas nesse capítulo as funções de transferência utilizadas para o projeto de controladores para dois tipos de movimentos do quadrirrotor.

Toda teoria necessária para compreender melhor a área de processamento de imagens é relatada no capítulo 4, desde aspectos básicos de imagens digitais até os algoritmos utilizados neste trabalho.

Os resultados são descritos no capítulo 5, no qual detalha os diferentes códigos produzidos para o processamento de imagens e comparações quantitativas dos resultados dos mesmos. Será abordado também o projeto dos controladores e o resultado de suas simulações bem como analisar o atendimento às especificações iniciais.

Por fim, conclusão e a apresentação de possíveis trabalhos futuros são descritos no capítulo 6, posteriormente tem-se as referências bibliográficas.

CAPÍTULO 2

SISTEMA

2.1 ASPECTOS GERAIS

Este capítulo tem por objetivo apresentar o quadrirrotor e todos os programas utilizados neste trabalho. Serão apresentadas as especificações técnicas do quadrirrotor, descrevendo o comportamento dinâmico do mesmo. O ambiente de programação e suas ferramentas serão detalhadas além de um aplicativo *mobile* utilizado para obtenção dos vídeos do AR.Drone.

2.2 DESCRIÇÃO DO QUADRIRROTOR AR.Drone

O quadrirrotor utilizado neste trabalho foi da marca *Parrot*, modelo *AR.Drone 1.0* [2], conforme mostrado na Figura 2.1.

Figura 2. 1 – Quadrirrotor *Parrot* AR.Drone[2] 1.0.

O Ar.Drone 1.0 foi apresentado pela primeira vez em 2010 na *International CES* (Mostra Internacional de Eletrônica de Consumo – em português) em Las Vegas através de uma demonstração de controle utilizando uma aplicação para smartphone *Apple*[3]. Após seu lançamento, o AR.Drone ficou bastante conhecido devido ao fácil manuseio e baixo custo comparado à sua gama área de aplicações [13].

[2] https://www.ifixit.com/Device/Parrot_AR.Drone
[3] http://en.wikipedia.org/wiki/Parrot_AR.Drone

2.2.1 COMPORTAMENTO DINÂMICO

O AR.Drone é um equipamento cuja estrutura mecânica consiste de 4 motores ligados a uma estrutura central em formato de X, conforme ilustrado na Figura 2.2. Na região central encontra-se toda a eletrônica e o *hardware* do sistema. Cada par de motores opostos giram no mesmo sentido, dessa forma é possível realizar todos os movimentos necessários. [2]

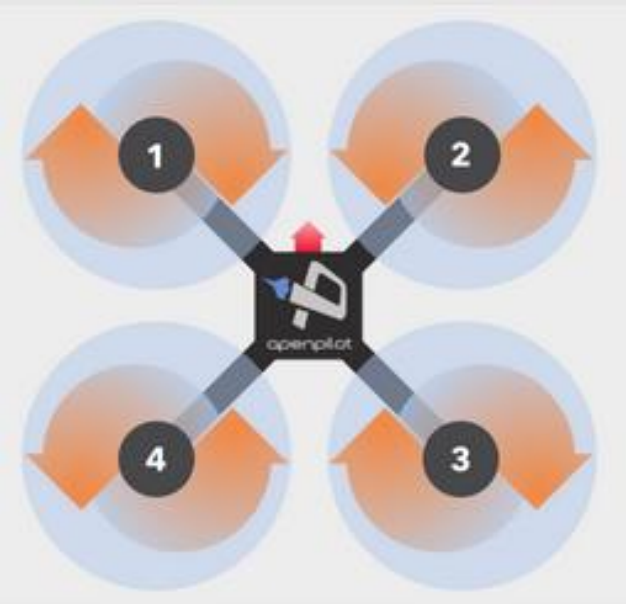

Figura 2. 2 – Estrutura mecânica do AR.Drone[4].

A movimentação do AR.Drone consiste basicamente na variação da velocidade e sentido dos motores (sempre considerando que motores opostos giram no mesmo sentido). São quatro tipos de movimentos possíveis, conforme ilustrado na Figura 2.3.

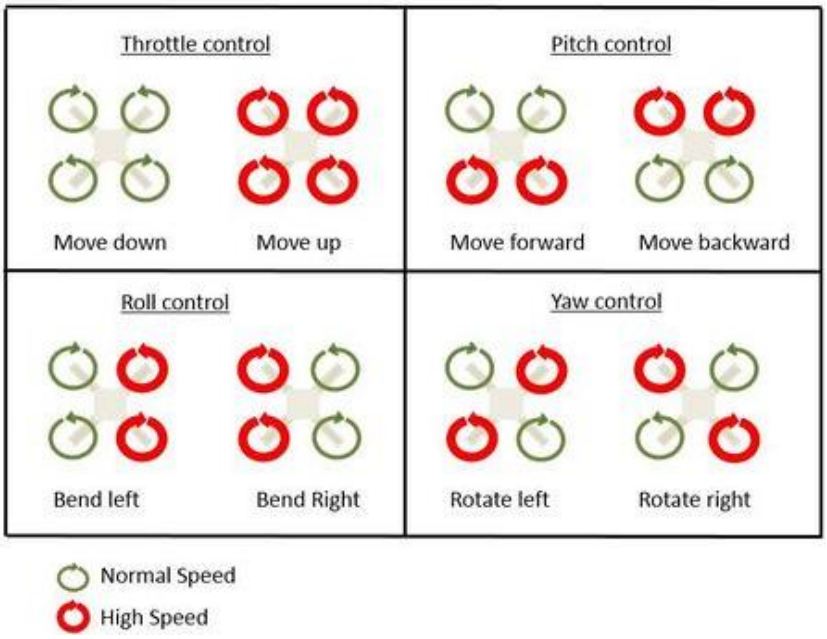

Figura 2. 3 – Movimentos existentes no AR.Drone[5].

[4] https://wiki.openpilot.org/display/WIKI/Vehicle

Os movimentos são realizados em torno de três eixos principais do corpo do quadrirrotor, conforme ilustrado na Figura 2.4. O movimento *throttle* (movimento de subida e descida) consiste no deslocamento no eixo Z; *pitch* (arfagem) é a rotação em relação ao eixo Y; *roll* (rolamento) é o mesmo movimento *pitch* só que em relação ao eixo X; por último o movimento *yaw* é o deslocamento no plano X-Y.

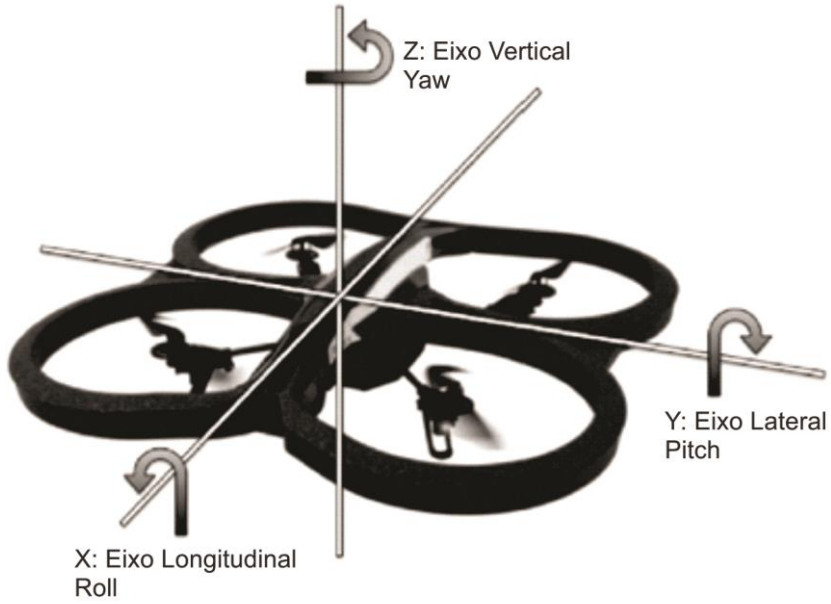

Figura 2. 4 – Eixos de atuação [17].

Os movimentos mostrados são efetuados através da variação dos ângulos nos respectivos eixos, exceto no movimento de *throttle,* em que não há rotação do corpo do quadrirrotor. O *pitch* (arfagem) é realizado na variação do ângulo θ (theta), *roll* (rolamento) na variação do ângulo *φ* (phi) e *yaw* na variação do ângulo *ψ* (psi). Todos os ângulos podem ser observados na Figura 2.5, além dos torques (Ti) e momentos (Mi) de cada motor.

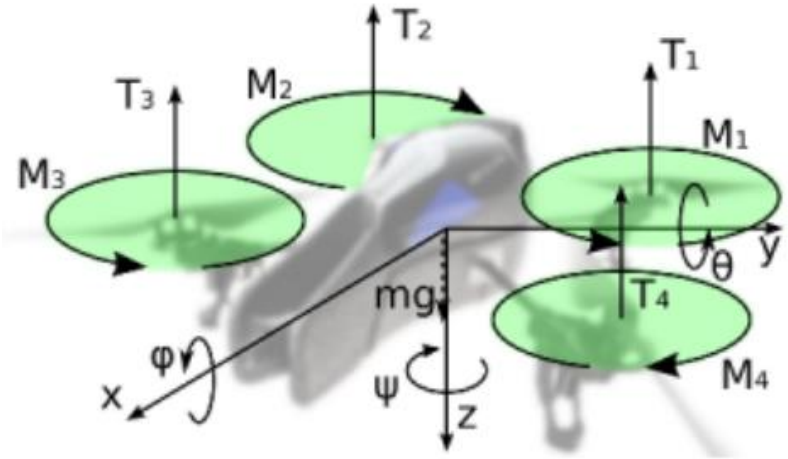

Figura 2. 5 – Variáveis do sistema, ângulos pertencentes a cada eixo[6].

[5] www.reddit.com
[6] http://www.decom.ufop.br/imobilis/?p=1254

As equações dinâmicas desses movimentos serão abordadas no Capítulo 3. Dois movimentos do AR.Drone foram amplamente abordados neste trabalho: *pitch* e *throttle*. Entretanto, o conceito é facilmente aplicável aos outros movimentos.

2.2.2 ESPECIFICAÇÕES TÉCNICAS

Os componentes pertencentes ao *hardware* do AR.Drone serão abordados nesta seção, onde as informações aqui contidas foram extraídas de [3] e [2].

Os principais componentes encontrados no *Parrot AR.Drone* são os seguintes:

- Câmera Frontal (QVGA) e Vertical (QCIF);
- três acelerômetros MEMS (Micro-Electro-Mechanical Systems);
- sensor ultrassônico de altitude;
- girômetro *MEMS* de dois eixos, movimento *pitch* (arfagem) e *roll* (rolamento);
- girômetro *MEMS* de precisão de um eixo, movimento *yaw*;
- bateria LiPo 1000mAh 11.1 V;
- quatro motores sem escova;
- CPU licenciado da ARM com Linux embarcado;
- comunicação digital via Wi-Fi b/g;

Os acelerômetros são sensores capazes de obter as acelerações lineares do quadrirrotor nos três eixos e os girômetros obtêm as acelerações angulares em torno dos três eixos. Esses sensores são definidos como *MEMS*, pois são construídos de forma otimizada espacialmente para embarcar em equipamentos com restrição de peso e tamanho.

A câmera vertical encontra-se acoplada junto à placa mãe do quadrirrotor, conforme ilustrado por um quadrado verde na Figura 2.6(a), a qual possui resolução QCIF (176 x 144 pixels) e ângulo de abertura de 64º. Já a câmera frontal possui resolução VGA (320 x 240 pixels), conforme Figura 2.6(b) e ângulo de abertura de 93º.

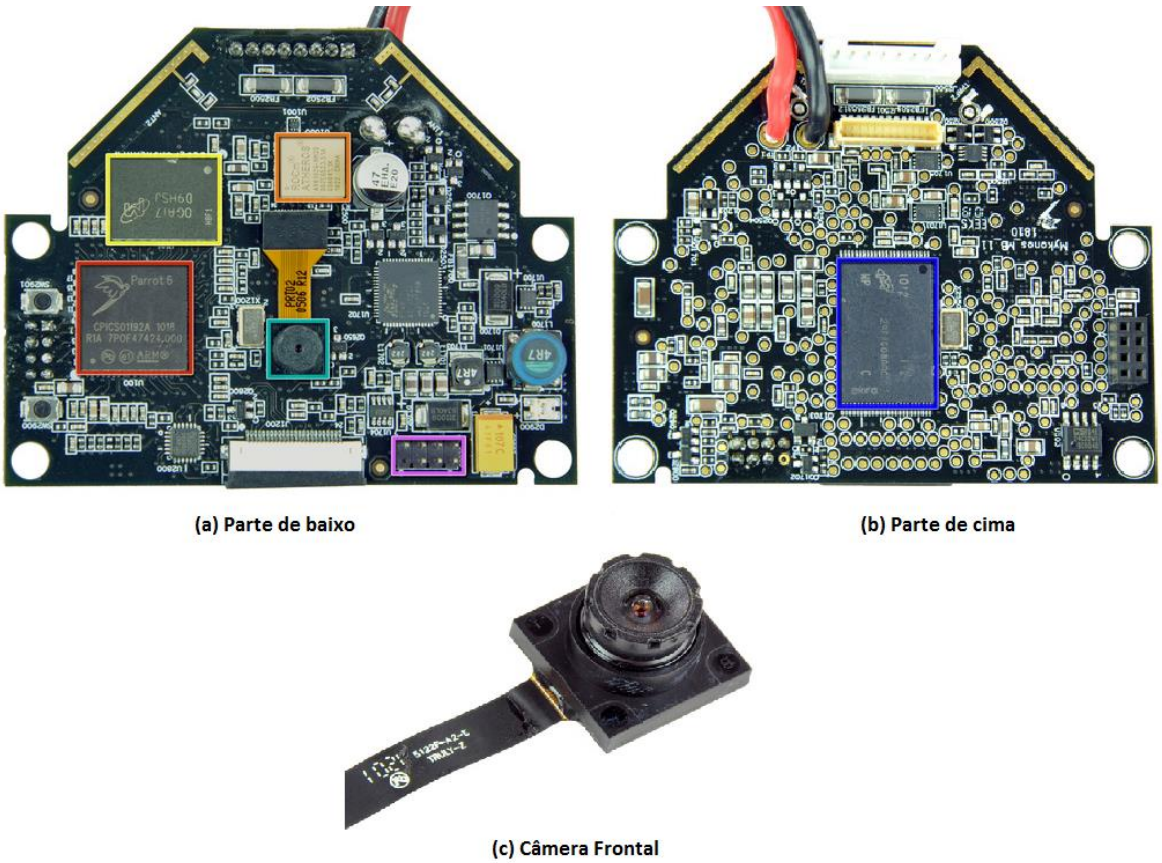

Figura 2. 6 – Placa mãe (a) e (b), câmera frontal (c).

A placa mãe do AR.Drone é o componente que se encontra o processamento central do quadrirrotor, sendo a CPU Parrot P6, com ARM926, no qual podem ser vistas as duas partes na Figura 2.6(a) e Figura 2.6(b). Diversos outros componentes são integrados junto à placa mãe, tal como o circuito integrado ROCm Atheros AR6102G-BM2D (laranja Figura 2.6(a)) usado na comunicação Wi-Fi nos padrões 802.11b e 802.11g. Outros dois circuitos integrados dentro da placa mãe são o Micron 29F1G08AAC (azul na Figura 2.6(b)) e o OGA17 D9HSJ (amarelo Figura 2.6(a)), conector USB/Serial (roxo na Figura 2.6(a)) e ainda as memórias flash e DDR SDRAM (ambas de 128 MB).

Os quatro motores são de alto desempenho com velocidade de 28.000 RPM quando “parado” (*Hover*) no ar até 41.400 RPM durante aceleração total. Cada motor possui seu próprio circuito controlador que inclui um microcontrolador, assim como um conversor ADC 10-bit, conforme ilustrado na Figura 2.7. O circuito controlador é responsável por manter a velocidade correta dos motores, além de desligá-los imediatamente em caso de travamento. As hélices foram projetadas de forma a obter consumo eficiente de energia e impulso necessário para o quadrirrotor.

(a) Vista superior (b) Vista inferior

Figura 2. 7 – Vista superior (a) e inferior (b) do conjunto motor, hélice e circuito controlador.

O sensor ultrassônico de altitude é composto por um cilindro emissor e outro receptor, os quais trabalham na frequência de 40khz, conforme ilustrado pela placa de navegação na Figura 2.8.

Figura 2. 8 – Sensor ultrassônico de altitude.

Na parte de cima da placa de navegação encontram-se o micro controlador PIC24HJ16GP304 (vermelho na Figura 2.9) responsável por todo o processamento dos dados dos sensores, os girômetros *Invensense IDG 500* (dois eixos) e *XV-3500CB* (um eixo) encontrados em laranja na Figura 2.9.

Figura 2. 9 – Micro controlador e girômetros do AR.Drone.

A bateria utilizada pelo AR.Drone é do tipo LiPo de 1000 mAh e tensão 11.1 V, que em condições normais possui autonomia de doze minutos de voo. A tensão da bateria é monitorada pela eletrônica do quadrirrotor, sendo que quando o nível se encontra baixo o mesmo pousa automaticamente. Este tipo de controle é necessário a fim de evitar um desligamento do controle em pleno voo, o que acarretaria em uma queda livre do aparelho.

2.3 PROGRAMAS UTILIZADOS

O programa principal utilizado neste trabalho foi o MATLAB, no qual foram realizadas diversas análises e simulações com o pacote SIMULINK. Dessa forma, foi possível de aplicar algoritmos de visão computacional e processamento de imagens, juntamente com projeto de controladores em um mesmo ambiente de programação.

2.3.1 MATLAB®

MATLAB[7] é um *software* interativo de fácil linguagem destinado à realização de cálculos e tarefas na área de engenharia, possuindo para isso uma ampla biblioteca de funções matemáticas. O nome vem da abreviação “MATrix LABoratory”, entretanto o mesmo é muito além do que um laboratório de matrizes.

[7] http://pt.wikipedia.org/wiki/MATLAB

Figura 2. 10 – Logotipo software MATLAB®[8].

O *software* da ferramenta foi desenvolvido por Clever Moler na década de 70, então presidente do departamento de Ciências da Computação da Universidade do Novo México. Por possuir uma linguagem muito semelhante à linguagem humana, foi bastante difundida nos ambientes acadêmicos principalmente na comunidade de engenharia e matemática. Jack Little, um engenheiro, conheceu o MATLAB em uma visita de Moler a Universidade de Stanford em 1983. Little se juntou a Moler e Steve Bangert para fundar a MathWorks, em 1984. Inicialmente, a linguagem foi difundida na área de projeto de controle, especialidade de Little, mas que rapidamente foi levada para outras áreas da engenharia.

O MATLAB possui diversos programas de apoio especializados, conhecidos como "*ToolBoxes*" que aumentam significativamente o número de funções que a linguagem é capaz de fornecer. As "*ToolBoxes*" estendem para praticamente todas as áreas da engenharia. Neste trabalho foi utilizado a *ToolBox* de processamento de imagens, o que possibilitou a análise e síntese dos dados aqui apresentados.

2.3.2 IMAGE PROCESSING TOOLBOX™

A Image Processing Toolbox[9] é um conjunto de algoritmos de apoio, funções e aplicativos que estabelecem variadas operações de processamento de imagens, tais como:

- Operações Morfológicas;
- síntese de filtros e filtragem de imagem linear;
- operações com transformadas;
- operações de segmentação;
- transformações geométricas;
- extração e descrição de características.

[8] http://www.palermo.edu/ingenieria/eventos/curso-matlab.html
[9] http://www.mathworks.com/products/image/

As principais operações de processamento de imagens utilizadas neste trabalho foram as operações morfológicas, extração e descrição de características e segmentação por cor, que serão abordados no capítulo 4.

2.3.3 *SIMULINK®*

O SIMULINK[10] é um pacote do MATLAB capaz de modelar, simular e analisar sistemas dinâmicos. Os sistemas analisados podem ser lineares ou não-lineares e ainda representados em tempo contínuo ou discreto.

Diferentemente do MATLAB, que utiliza linhas de comando para criar funções e algoritmos, o SIMULINK é baseado em diagramas de blocos que podem tornar o projeto mais intuitivo por se tratar de uma abordagem gráfica. Fácil de ser utilizado o SIMULINK possui bibliotecas de blocos pré-definidos, nos quais é só definir os blocos e montar o projeto como um todo.

Apesar de ser uma aplicação específica, o SIMULINK não funciona independente do MATLAB, os resultados das simulações realizadas no SIMULINK podem ser visualizados sendo enviados ao *workspace* do MATLAB, dessa forma, a integração dos dois é de fundamental importância.

Neste trabalho foi utilizado como base, para o projeto de controladores do AR.Drone, uma aplicação desenvolvida no SIMULINK, que permite a comunicação, simulação e visualização dos estados em tempo real do quadrirrotor. O detalhamento desta aplicação pode ser visto na próxima seção.

2.3.4 AR DRONE SIMULINK DEVELOPMENT KIT[11]

O kit de desenvolvimento consiste de blocos no SIMULINK que estabelecem a possibilidade de conexão em tempo real via Wi-Fi e simulação da dinâmica do AR.Drone Parrot. A aplicação foi desenvolvida por David Escobar Sanabria e Pieter J. Mosterman no contexto do projeto de estágio de verão da MathWorks em 2013 [20].

Os blocos de simulação são baseados em modelos obtidos via identificação de sistemas utilizando um AR.Drone real. Os modelos são divididos nos diversos movimentos possíveis do quadrirrotor, *yaw, pitch, roll* e *throttle*.

Os blocos de controle estabelecem conexão via Wi-Fi, podendo enviar comandos, tendo como contrapartida a leitura dos estados do quadrirrotor em tempo real. A comunicação é realizada por canais *udp* de troca de mensagens.

[10] http://www.enautica.pt/publico/professores/baptista/instrum/Intro_Simulink.pdf
[11] http://www.mathworks.com/matlabcentral/fileexchange/43719-ar-drone-simulink-development-kit-v1-

Tanto para os blocos de simulação quanto para o controle via Wi-Fi é possível realizar movimentos em cima de trajetórias ou simplesmente cada movimento em separado por vez. A simulação é composta por três blocos principais: (a) bloco cinza, que consiste do controle básico (proporcional unitário) para cada estado, (b) bloco amarelo que contém o modelo do quadrirrotor e estabelece a dinâmica do movimento e (c) o bloco verde que estabelece uma estimação mais precisa da posição do quadrirrotor [20]. Esses blocos podem ser observados na Figura 2.11.

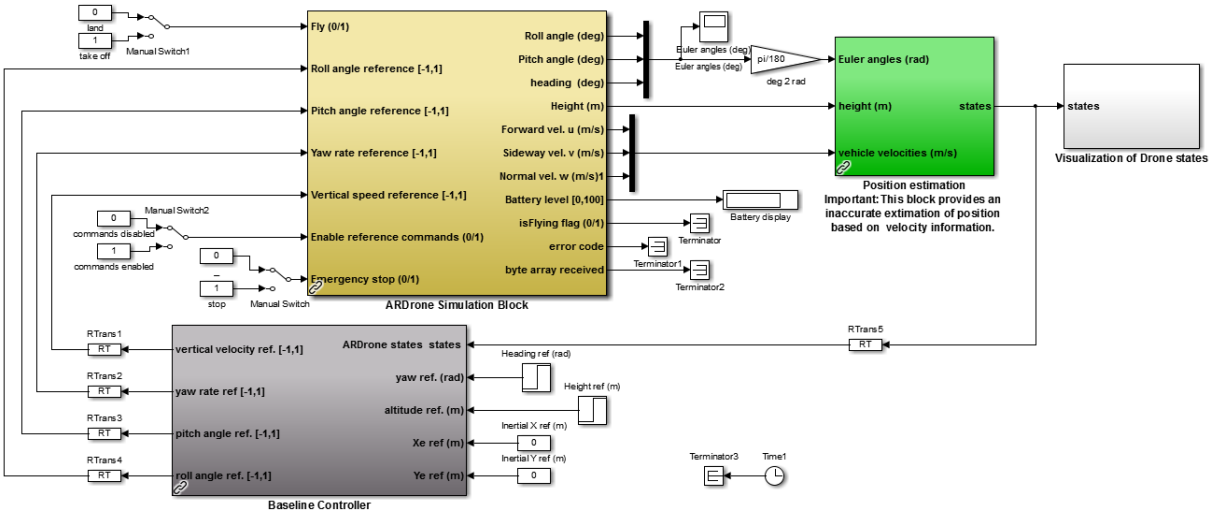

Figura 2. 11 – Blocos de simulação do AR.Drone no SIMULINK® [20].

2.3.5 APLICATIVO ANDROID AR.FreeFlight[12]

Durante a elaboração deste trabalho foi utilizado um aplicativo mobile para sistema Android, chamado AR.FreeFlight, capaz de controlar o AR.Drone, comportando-se de maneira semelhante a um controle. Com este aplicativo é possível controlar o quadrirrotor tendo acesso aos dados dos sensores, tais como a altura e velocidade de deslocamento frontal. Além disso, é possível visualizar em tempo real as imagens provenientes das câmeras, das quais é possível gravar vídeos ou simplesmente tirar fotos. A tela principal de controle do AR.Drone pode ser visualizada pela Figura 2.12.

[12] https://play.google.com/store/apps/details?id=com.parrot.freeflight&hl=pt_BR

Figura 2. 12 – Tela de navegação do AR.Drone via app Android.

Todos os algoritmos de processamento de imagem foram aplicados nos vídeos capturados pelo aplicativo. Além disso, os dados da altura foram confrontados com a medição via fita métrica, a fim de realizar a relação mm/pixel que será mostrada no Capítulo 4.

CAPÍTULO 3

MODELO

3.1 ASPECTOS GERAIS

Neste capítulo será apresentado o equacionamento do modelo teórico de um quadrirrotor e do conjunto motor/hélice. Adicionalmente, serão mostradas as funções de transferência extraídas do kit de Desenvolvimento do MatLab realizado no Simulink [20].

3.2 MODELO TEÓRICO DE UM QUADRIRROTOR

O modelo teórico de um quadrirrotor é descrito a partir de equações de um corpo rígido genérico com seis graus de liberdade baseados no modelo de Newton-Euler descritos em [15].

Neste caso, definem-se dois tipos de referência, a referência terrestre (sistema E) e a referência do corpo rígido do quadrirrotor (sistema B), conforme mostrado na Figura 3.1.

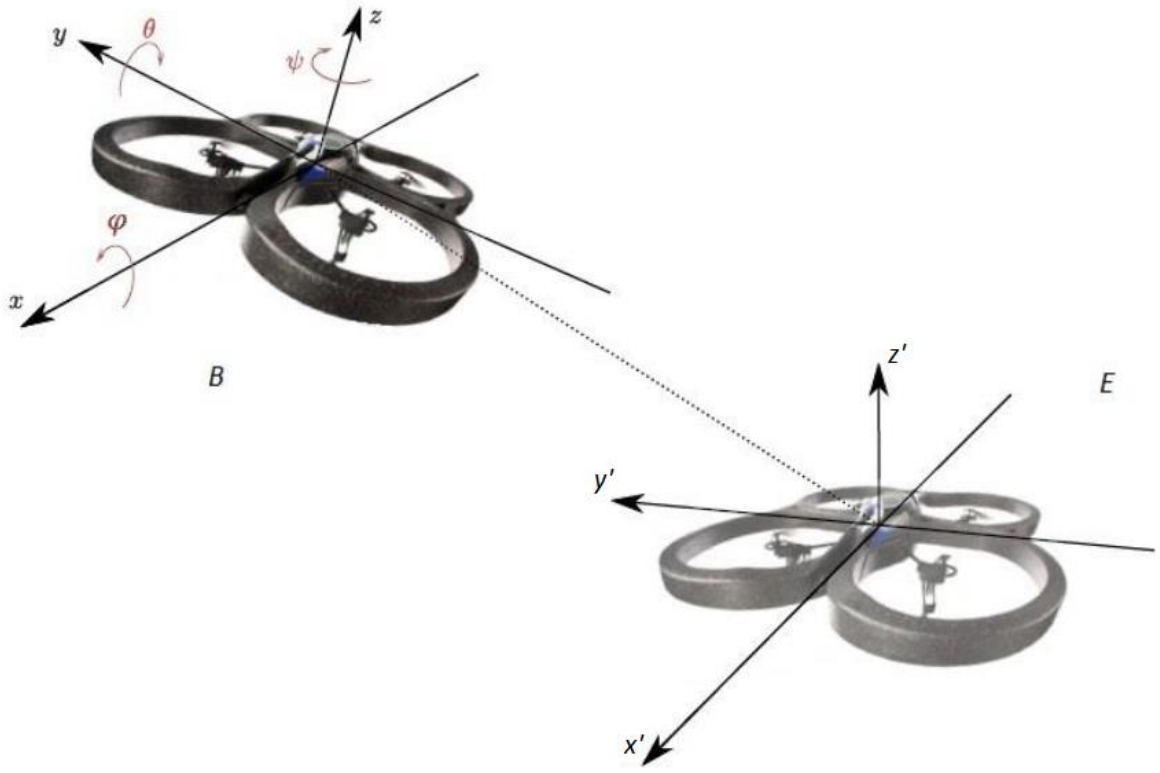

Figura 3. 1 – Sistema de referência terrestre (E) e do corpo rígido (B), editada de [3].

As equações de movimento são mais convenientemente definidas utilizando a referência do corpo fixo do quadrirrotor pelas seguintes razões:

- A matriz de inércia é invariante no tempo;
- a simetria do quadrirrotor tem a vantagem de simplificar as equações;
- as medições de sensores do quadrirrotor são mais fáceis de serem convertidas para a referência do corpo rígido;
- as forças responsáveis pelo controle são dadas na referência do corpo rígido.

Primeiramente é apresentada a equação (3.1) que descreve a cinemática de um corpo rígido de seis graus de liberdade [15].

$$\dot{\boldsymbol{\zeta}} = \boldsymbol{J}_{\Theta}\boldsymbol{\nu}, \tag{3.1}$$

no qual $\dot{\boldsymbol{\zeta}}$ é o vetor de velocidade em relação ao sistema E, $\boldsymbol{\nu}$ é o vetor de velocidade em relação ao sistema B e $\boldsymbol{J}_{\Theta}$ é a matriz generalizada.

Assim, $\boldsymbol{\zeta}$ é constituído dos vetores de posições linear $\Gamma^{\boldsymbol{E}}$ [m] e angular $\boldsymbol{\Theta}^{\boldsymbol{E}}$ [rad] do quadrirrotor no sistema E, conforme mostrado na equação (3.2).

$$\boldsymbol{\zeta} = [\Gamma^{\boldsymbol{E}} \quad \boldsymbol{\Theta}^{\boldsymbol{E}}]^{\boldsymbol{T}} = [X \quad Y \quad Z \quad \varphi \quad \theta \quad \psi]^{\boldsymbol{T}} \tag{3.2}$$

Da mesma forma, $\boldsymbol{\nu}$ é constituído dos vetores de velocidades linear $\boldsymbol{V}^{\boldsymbol{B}}$ [m/s] e angular $\boldsymbol{\omega}^{\boldsymbol{B}}$ [rad/s] do quadrirrotor no sistema B, conforme mostrado na equação (3.3).

$$\boldsymbol{\nu} = [\boldsymbol{V}^{\boldsymbol{B}} \quad \boldsymbol{\omega}^{\boldsymbol{B}}]^{\boldsymbol{T}} = [u \quad v \quad w \quad p \quad q \quad r]^{\boldsymbol{T}} \tag{3.3}$$

Por fim, a matriz generalizada é composta por 4 sub-matrizes, conforme a equação (3.4),

$$\boldsymbol{J}_{\Theta} = \begin{bmatrix} \boldsymbol{R}_{\Theta} & \boldsymbol{0_{3x3}} \\ \boldsymbol{0_{3x3}} & \boldsymbol{T}_{\Theta} \end{bmatrix}, \tag{3.4}$$

onde $\boldsymbol{0_{3x3}}$ significa uma matriz quadrada de dimensão 3x3 com elementos nulos e as matrizes de rotação $\boldsymbol{R}_{\Theta}$ e translação $\boldsymbol{T}_{\Theta}$ são definidas pelas equações (3.5) e (3.6) [15].

$$\boldsymbol{R}_{\Theta} = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\varphi + c_\psi s_\theta s_\varphi & s_\psi s_\varphi + c_\psi s_\theta c_\varphi \\ s_\psi c_\theta & c_\psi c_\varphi + s_\psi s_\theta s_\varphi & -c_\psi s_\varphi + s_\psi s_\theta c_\varphi \\ -s_\theta & c_\theta s_\varphi & c_\theta c_\varphi \end{bmatrix} \tag{3.5}$$

$$\boldsymbol{T}_{\Theta} = \begin{bmatrix} 1 & s_\varphi t_\theta & c_\varphi t_\theta \\ 0 & c_\varphi & -s_\varphi \\ 0 & {s_\varphi}/{c_\theta} & {c_\varphi}/{c_\theta} \end{bmatrix} \tag{3.6}$$

Para minimizar a leitura considerou-se a seguinte definição usada nas equações (3.5) e (3.6): $c_k = cos(k)$, $s_k = sen(k)$, $t_k = tan(k)$.

A dinâmica do sistema então pode ser descrita como a equação (3.7) (conforme [15]) considerando *m* [kg] como a massa do corpo e I [N*m*s²] como a matriz de inércia.

$$\begin{bmatrix} m\mathrm{I}_{3x3} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathrm{I} \end{bmatrix}\begin{bmatrix} \dot{\boldsymbol{V}}^{\boldsymbol{B}} \\ \dot{\boldsymbol{\omega}}^{\boldsymbol{B}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\omega}^{\boldsymbol{B}} \boldsymbol{x} \left(m\boldsymbol{V}^{\boldsymbol{B}}\right) \\ \boldsymbol{\omega}^{\boldsymbol{B}} \boldsymbol{x} \left(\mathrm{I}\boldsymbol{\omega}^{\boldsymbol{B}}\right) \end{bmatrix} = \begin{bmatrix} \boldsymbol{F}^{\boldsymbol{B}} \\ \boldsymbol{\tau}^{\boldsymbol{B}} \end{bmatrix}, \tag{3.7}$$

no qual I_{3x3} é a matriz identidade de dimensão 3x3, $\dot{\boldsymbol{V}}^{\boldsymbol{B}}$ [m/s²] é o vetor aceleração linear do quadrirrotor, $\dot{\boldsymbol{\omega}}^{\boldsymbol{B}}$ [rad/s²] é o vetor aceleração angular. $\boldsymbol{F}^{\boldsymbol{B}}$ [N] é o vetor de forças e $\boldsymbol{\tau}^{\boldsymbol{B}}$ [N*m] é o vetor de torques. Todas as variáveis na referência do sistema B.

O vetor de força generalizada é definido como $\boldsymbol{\Lambda}$, de acordo com a equação (3.8).

$$\boldsymbol{\Lambda} = [\boldsymbol{F}^{\boldsymbol{B}} \quad \boldsymbol{\tau}^{\boldsymbol{B}}]^{\mathrm{T}} = [F_x \quad F_y \quad F_z \quad \tau_{\boldsymbol{x}} \quad \tau_{\boldsymbol{y}} \quad \tau_{\boldsymbol{z}}]^{\mathrm{T}}. \tag{3.8}$$

Assim a equação (3.7) pode ser reescrita conforme a equação (3.9).

$$\boldsymbol{M}_{\boldsymbol{B}}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{\boldsymbol{B}}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\Lambda}, \tag{3.9}$$

onde $\dot{\boldsymbol{\nu}}$ é o vetor generalizado das acelerações, $\boldsymbol{M}_{\boldsymbol{B}}$ é a matriz de inércia do sistema e $\boldsymbol{C}_{\boldsymbol{B}}$ é a matriz de aceleração centrípeta de Coriolis, todos na referência do sistema B [15]. $\boldsymbol{M}_{\boldsymbol{B}}$ e $\boldsymbol{C}_{\boldsymbol{B}}$ são mostradas nas equações (3.10) e (3.11):

$$\boldsymbol{M}_{\boldsymbol{B}} = \begin{bmatrix} \boldsymbol{m}\mathrm{I}_{3x3} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathrm{I} \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathrm{I}_{XX} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathrm{I}_{YY} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathrm{I}_{ZZ} \end{bmatrix} \tag{3.10}$$

$$\boldsymbol{C}_{\boldsymbol{B}}(\boldsymbol{\nu}) = \begin{bmatrix} \mathbf{0}_{3x3} & -m(\boldsymbol{V}^{\boldsymbol{B}}) \\ \mathbf{0}_{3x3} & -(\mathrm{I}\boldsymbol{\omega}^{\boldsymbol{B}}) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & mw & -mv \\ 0 & 0 & 0 & -mw & 0 & mu \\ 0 & 0 & 0 & mv & -mu & 0 \\ 0 & 0 & 0 & 0 & \mathrm{I}_{ZZ}r & -\mathrm{I}_{YY}q \\ 0 & 0 & 0 & -\mathrm{I}_{ZZ}r & 0 & \mathrm{I}_{XX}p \\ 0 & 0 & 0 & \mathrm{I}_{YY}q & \mathrm{I}_{XX}p & 0 \end{bmatrix}. \tag{3.11}$$

A equação (3.9) é genérica para qualquer corpo rígido, considerando a aplicação para um quadrirrotor. Conforme detalhado no capítulo 2 temos que o vetor $\boldsymbol{\Lambda}$ é dividido em três componentes.

A primeira componente é a contribuição da aceleração da gravidade g [m/s²], representada pelo vetor gravitacional $\boldsymbol{G}_{\boldsymbol{B}}(\boldsymbol{\zeta})$**,** dado pela equação (3.12) (segundo [15]):

$$\boldsymbol{G}_{\boldsymbol{B}}(\boldsymbol{\zeta}) = \begin{bmatrix} \boldsymbol{F}_{\boldsymbol{G}}^{\boldsymbol{B}} \\ \mathbf{0}_{3x1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\theta}^{-1}\boldsymbol{F}_{\boldsymbol{G}}^{\boldsymbol{E}} \\ \mathbf{0}_{3x1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{\theta}^{\boldsymbol{T}} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ -m\boldsymbol{g} \end{bmatrix} \\ \mathbf{0}_{3x1} \end{bmatrix} = \begin{bmatrix} m\boldsymbol{g}s_{\theta} \\ -m\boldsymbol{g}c_{\theta}s_{\varphi} \\ -m\boldsymbol{g}c_{\theta}s_{\varphi} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{3.12}$$

onde $\boldsymbol{F}_{\boldsymbol{G}}^{\boldsymbol{B}}$ e $\boldsymbol{F}_{\boldsymbol{G}}^{\boldsymbol{E}}$ são os vetores de força gravitacional no sistema B e E respectivamente.

A segunda contribuição se refere aos efeitos giroscópicos causados pela diferença de rotação dos rotores, que, quando a soma algébrica não é igual a zero há um não

balanceamento e o torque resultante no quadrirrotor, quando os ângulos de rolamento e arfagem são diferentes de zero, e de acordo com a equação (3.13).

$$\boldsymbol{O_B}(\boldsymbol{\nu})\Omega = \begin{bmatrix} \mathbf{0_{3x1}} \\ -\sum_{\boldsymbol{k=1}}^{\boldsymbol{4}} \boldsymbol{J_{TS}} \left(\boldsymbol{\omega^B} x \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) (-\mathbf{1})^{\boldsymbol{k}} \Omega_k \end{bmatrix} \tag{3.13}$$

$$= \begin{bmatrix} \mathbf{0_{3x1}} \\ \boldsymbol{J_{TS}} \begin{bmatrix} -\boldsymbol{q} \\ \boldsymbol{p} \\ \mathbf{0} \end{bmatrix} \Omega \end{bmatrix} = \boldsymbol{J_{TS}} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ q & -q & q & -q \\ -p & p & -p & p \\ 0 & 0 & 0 & 0 \end{bmatrix} \Omega$$

A terceira contribuição é derivada das forças e torques aplicados diretamente ao quadrirrotor através dos motores, o que resulta em um vetor de movimento denominado $\boldsymbol{U_B}(\Omega)$, conforme indicado na equação (3.14) (vide [15]):

$$\boldsymbol{U_B}(\Omega) = \boldsymbol{E_B}\Omega^2 = \begin{bmatrix} 0 \\ 0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ b(\Omega_{\mathbf{1}}^2 + \Omega_{\mathbf{2}}^2 + \Omega_{\mathbf{3}}^2 + \Omega_{\mathbf{4}}{}^2) \\ bl(\Omega_{\mathbf{4}}^2 - \Omega_{\mathbf{2}}{}^2) \\ bl(\Omega_{\mathbf{3}}^2 - \Omega_{\mathbf{1}}{}^2) \\ d(\Omega_{\mathbf{2}}^2 + \Omega_{\mathbf{4}}^2 - \Omega_{\mathbf{1}}^2 - \Omega_{\mathbf{3}}{}^2) \end{bmatrix}, \tag{3.14}$$

onde $\boldsymbol{E_B}$ é a matriz de movimento, b [N*s²] é o coeficiente de empuxo, d [N*m*s²] é o coeficiente de arrasto e l [m] é a distância entre o centro do quadrirrotor e o centro dos motores.

Reescrevendo a equação (3.9), pode-se obter a seguinte equação da dinâmica do sistema (vide equação (3.15)):

$$\boldsymbol{M_B}\dot{\boldsymbol{\nu}} + \boldsymbol{C_B}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{G_B}(\boldsymbol{\zeta}) + \boldsymbol{O_B}(\boldsymbol{\nu})\Omega + \boldsymbol{E_B}\Omega^2. \tag{3.15}$$

Isolando a derivada do vetor velocidade, obtém-se:

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M_B^{-1}}\left(-\boldsymbol{C_B}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{G_B}(\boldsymbol{\zeta}) + \boldsymbol{O_B}(\boldsymbol{\nu})\Omega + \boldsymbol{E_B}\boldsymbol{\Omega^2}\right). \tag{3.16}$$

A representação da equação (3.16) em forma de sistema de equações é mostrada a seguir:

$$\begin{cases} \dot{u} = (vr - wq) + gs\theta \\ \dot{v} = (wp - ur) - gc_\theta s_\varphi \\ \dot{w} = (uq - vp) - gc_\theta s_\varphi + \dfrac{U_1}{m} \\ \dot{p} = \dfrac{I_{YY} - I_{ZZ}}{I_{XX}} qr - \dfrac{J_{TS}}{I_{XX}} q\Omega + \dfrac{U_2}{I_{XX}} \\ \dot{q} = \dfrac{I_{ZZ} - I_{XX}}{I_{YY}} pr + \dfrac{J_{TS}}{I_{YY}} p\Omega + \dfrac{U_3}{I_{YY}} \\ \dot{r} = \dfrac{I_{XX} - I_{YY}}{I_{ZZ}} pq + \dfrac{U_4}{I_{ZZ}} \end{cases} \tag{3.17}$$

A velocidade das hélices pode ser extraída também e apresentada pela equação (3.18).

$$\begin{cases} U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4{}^2) \\ U_2 = bl(\Omega_4^2 - \Omega_2{}^2) \\ U_3 = bl(\Omega_3^2 - \Omega_1{}^2) \\ U_4 = d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3{}^2) \\ \Omega = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{cases} \quad (3.18)$$

Segundo a modelagem proposta em [15] (e adotado neste trabalho) o modelo de um quadrirrotor é não linear, e que suas variáveis de estado são dependentes entre si. Entretanto, de acordo com [14] e [16], nos limites de voo de um AR.Drone (ângulo de atuação menor que 10º para arfagem e rolamento) as equações de movimento são representadas por sistemas lineares, isso quer dizer que não precisamos tratar o AR.Drone como um sistema único controlado apenas não-linearmente.

Além disso, em [14] o AR.Drone é comparado à uma caixa preta (Figura 3.2) no qual o que o que se tem acesso são apenas as variáveis de entrada e saída. Por outro lado, o próprio sistema possui um controle interno o qual não se tem acesso.

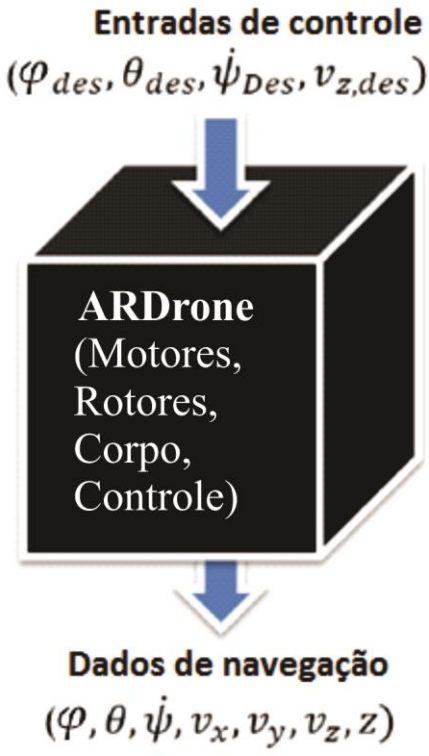

Figura 3. 2 – Representação do AR.Drone como uma caixa preta, adaptado de [14].

3.3 LINEARIZAÇÃO DAS EQUAÇÕES DE VELOCIDADE, ADOTADA NESTE TRABALHO

Para este trabalho as principais equações são as três primeiras (da equação (3.17), *u*, *v* e *w*) que são as acelerações de translação do quadrirrotor. Tendo em conta que em um movimento de *pitch* as variáveis de *roll* e *throttle* não interferem no movimento, considerar-se-á as mesmas nulas. Dessa forma neste trabalho foi calculada a aceleração $\dot{u}$, segundo a expressão (3.19):

$$\dot{u} = g * sin(\theta)\,. \tag{3.19}$$

Isolando o θ, temos:

$$\theta = \sin^{-1}\frac{\dot{u}}{g}. \tag{3.20}$$

Isso mostra que o ângulo de arfagem dependerá apenas da aceleração obtida pelo acelerômetro do AR.Drone e que para pequenos movimentos (ângulo de arfagem menor que 0.17 rad ou 10º) a equação pode ser aproximada por:

$$\theta = \frac{\dot{u}}{g}. \tag{3.21}$$

Da mesma forma, a aceleração vertical no movimento de *throttle,* $\dot{w}$, pode ser aproximada, considerando todas as outras componentes nulas, por:

$$\dot{w} = \frac{U_1}{m}. \tag{3.22}$$

A aceleração $\dot{w}$ só depende então da força resultante dos motores em relação à massa do corpo. Os modelos apresentados neste trabalho levaram em conta as independências das variáveis de estado, o que permitiu projetar controladores lineares para tais movimentos.

3.4 MODELO DO CONJUNTO MOTOR/HÉLICE

A configuração que aqui será apresentada serve de base para entender a conexão de um motor com a hélice nele ligada. A modelagem considerando as ligações das engrenagens, e consequente passagem de torque, é de suma importância para definir qual é a relação da tensão de entrada em um motor com a velocidade da hélice, que aplica a propulsão necessária para movimentar o quadrirrotor. Para este trabalho foi novamente adoptado os modelos propostos em [15].

Neste caso, o atuador apresentado é o motor de corrente contínua (DC) pelo qual a atuação é feita pela conversão de energia elétrica por energia mecânica (movimento). O

modelo do motor é composto por três parâmetros em série, R $[\Omega]$ o resistor, L $[H]$ o indutor e uma fonte de tensão conhecida como força eletromotriz (f.e.m) e $[V]$. A representação do modelo de um motor DC é mostrada na Figura 3.3.

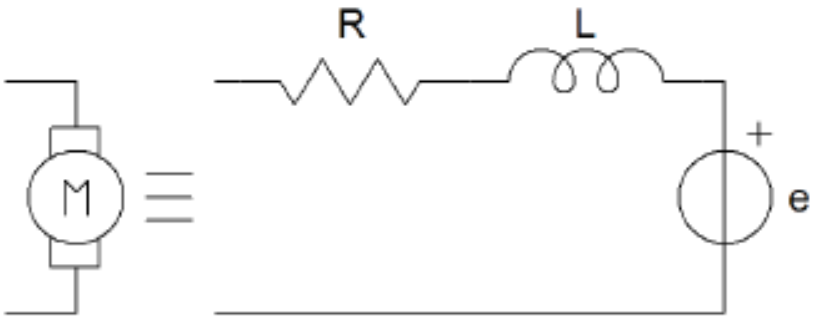

Figura 3. 3 – Circuito representativo de um motor elétrico.

Aplicando a lei das tensões de Kirchhoff, obtemos a seguinte expressão:

$$v_t = v_R + v_L + e\,. \tag{3.23}$$

A equação (3.23) pode ser reescrita da seguinte forma:

$$v_t = Ri + L\frac{di}{dt} + K_E w_M\,, \tag{3.24}$$

onde i $[A]$ é a corrente que passa no motor, K_E $[Vs/rad]$ é a constante do motor e w_M $[rad/s]$ é a velocidade angular do motor [15].

A indutância de motores aplicados à robótica é geralmente muito pequena, se comparada à contribuição mecânica do sistema, dessa forma, a equação (3.24) é reescrita da seguinte forma:

$$v_t = Ri + K_E w_M\,. \tag{3.25}$$

A dinâmica do sistema motor/carga pode ser escrita como:

$$J_{TM}\dot{w}_M = T_M - T_L\,, \tag{3.26}$$

onde J_{TM} $[N * m * s^2]$ é o momento de inércia total do motor, T_M $[N * m]$ é o torque do motor e T_L $[N * m]$ é o torque da carga, conforme mostrado na Figura 3.4.

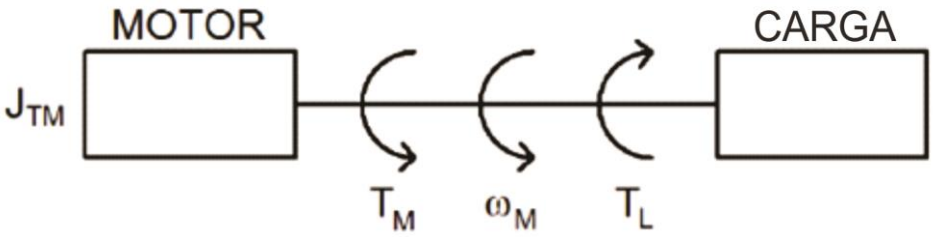

Figura 3. 4 – Acoplamento e transferência de energia do motor para a carga.

No quadrirrotor a carga é o conjunto engrenagem/hélice, conforme mostrado na Figura 3.5.

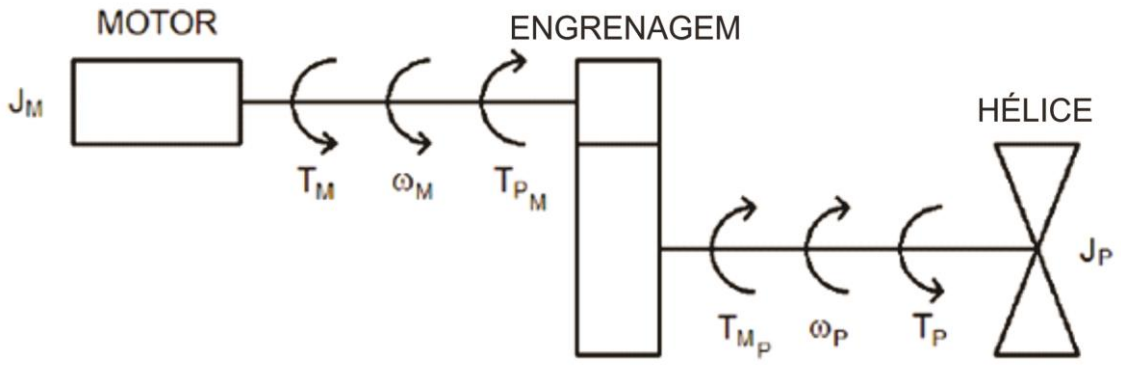

Figura 3. 5 – Conjunto motor, engrenagem e hélice.

onde J_P $[N * m * s^2]$ é o momento de inércia da hélice em relação ao seu eixo, T_{P_M} $[N * m]$ é o torque da hélice aplicado ao eixo do motor, T_{M_P} $[N * m]$ é o torque do motor aplicado ao eixo da hélice e T_P $[N * m]$ é o torque da hélice [15].

O equacionamento final do sistema relaciona a tensão de entrada do motor com a aceleração angular da hélice, conforme mostrado na equação (3.27):

$$\dot{w}_P = A_P \mathrm{w}_P - B_P v_t + C_P \,, \tag{3.27}$$

A_P $[rad/s]$ é o coeficiente de velocidade da hélice, B_P $[rad^2/s^2V]$ é o coeficiente da tensão de entrada do motor e C_P $[rad^2/s^2]$ é o coeficiente constante, todos são definidos como mostrado abaixo:

$$A_P = -\frac{K_E K_M}{R J_{TP}} \eta N^2$$

$$B_P = \frac{K_M}{R J_{TP}} \eta N \tag{3.28}$$

$$C_P = -\frac{d\mathrm{w}^2{}_P}{J_{TP}},$$

onde J_{TP} $[N * m * s^2]$ é o momento de inércia rotacional total em relação ao eixo da hélice, η é a eficiência da conversão em relação à potência do motor transmitida para a hélice, $N = \frac{\mathrm{w}_M}{\mathrm{w}_P}$ é a razão de redução da engrenagem e K_M é a constante de proporcionalidade do torque do motor em relação à corrente que atravessa o mesmo.

O vetor aceleração das quatro hélices e o vetor das tensões de entrada nos quatro motores são relacionados como:

$$\dot{\Omega}_{\boldsymbol{P}} = A_P \Omega_{\boldsymbol{P}} - B_P \boldsymbol{v} + C_P \tag{3.29}$$

As equações aqui mostradas são apenas de cunho demonstrativo, onde as funções de transferência do sistema para simulação e projeto de controladores foram obtidas por David Escobar Sanabria e Pieter J. Mosterman [20].

3.5 FUNÇÃO DE TRANSFERÊNCIA DO SISTEMA

No projeto foi determinado que o controle de trajetória fosse realizado nas orientações das coordenadas X, Y e Z, ou seja, o ângulo de arfagem, o ângulo de rolamento e a altura do quadrirrotor. As funções de transferência foram obtidas através do kit de desenvolvimento no Simulink abordado na Seção 2.3.4, cuja identificação foi realizada por David Escobar Sanabria e Pieter J. Mosterman [20].

Vale ressaltar ainda que o período de amostragem do sistema é de 0.065 segundos, um parâmetro bastante importante no projeto dos controladores que será abordado no capítulo 5.

3.5.1 FUNÇÃO ALTURA DO QUADRIRROTOR

A função de transferência para a dinâmica do movimento no eixo da altura (Z) do quadrirrotor tem como entrada a altura de referência pelo qual o quadrirrotor deve subir (ou descer) e a saída é posição da altura em que o quadrirrotor se movimentou após responder ao comando.

Na seção 3.3 foi relatado que o modelo experimental do AR.Drone, intrinsecamente não linear, poderia ser representado por uma função linear desde que dentro dos limites de atuação dos movimentos, como ângulo de arfagem menor que 10º. Os modelos encontrados no kit de desenvolvimento no SIMULINK® são todos lineares e independentes. A função de transferência no tempo discreto em malha aberta do deslocamento da altura do AR.Drone, com tempo de amostragem igual a 0.065s, é mostrado na equação (3.30),

$$Z_a = \frac{0.01789z + 0.000232}{z^2 - 1.685z + 0.685}. \qquad (3.30)$$

Para tanto, foi realizada simulação da dinâmica do modelo em malha fechada, conforme mostrado na Figura 3.6 do *SIMULINK*®.

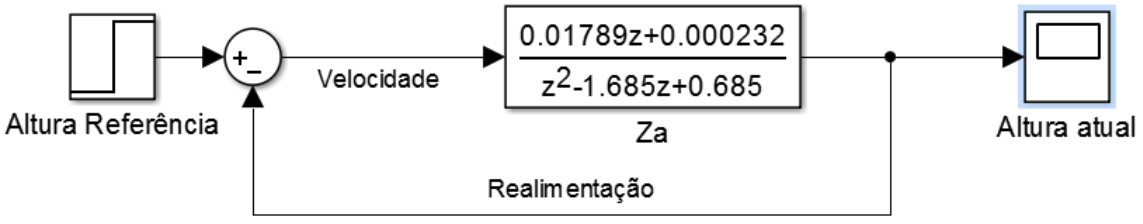

Figura 3. 6 – Diagrama de blocos em malha fechada da altura sem controlador.

É possível observar pela Figura 3.6 que a diferença (denominada erro) entre a *Altura Referência* e a *Altura Atual* nada mais é que a velocidade pela qual o AR.Drone necessita movimentar até atingir a referência pois quando isso acontece o valor do comando para a velocidade é zero.

Neste caso, foi realizada a simulação com uma *Altura Referência* como um degrau de 1 m a partir do primeiro segundo e mantendo esse valor indefinidamente. A resposta apresentada pode ser observada pela Figura 3.7.

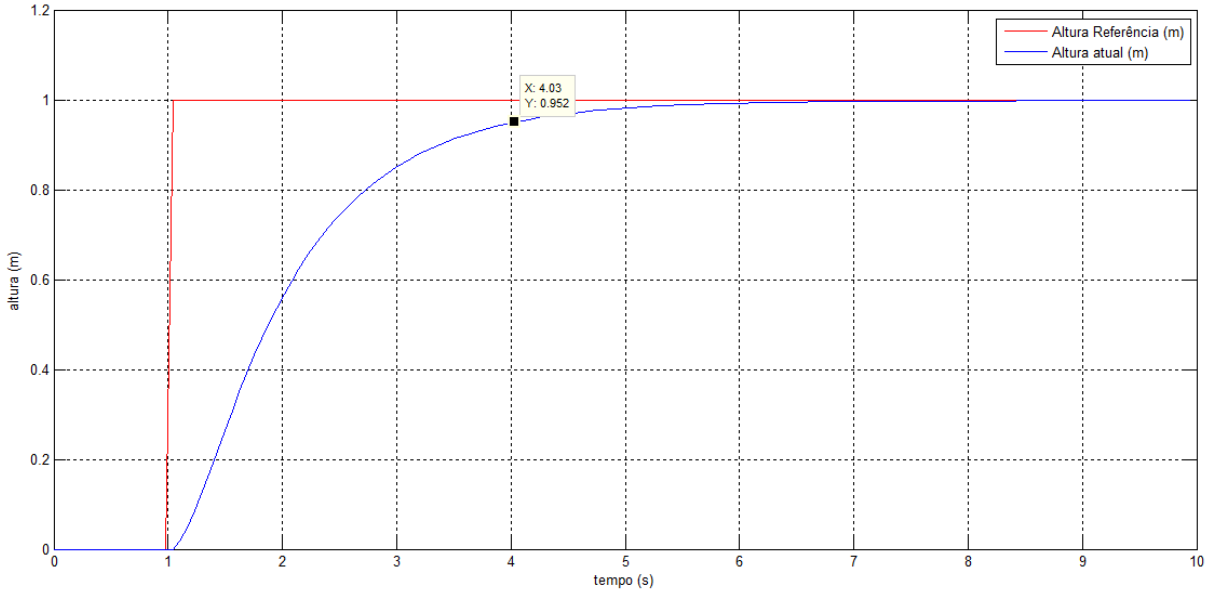

Figura 3. 7 – Resposta da altura com entrada degrau.

Conforme ilustrado na Figura 3.7 a resposta da altura em malha fechada sem controlador indica um tempo de resposta de 3.03 segundos, sem sobressinal e erro estacionário nulo. Esse tempo de resposta é obtido observando o momento em que a resposta atinge um *range* de 5% do valor em regime permanente. O sistema responde lentamente ao comando de subida, mas alcança a referência em um determinado momento. Essa dinâmica pode ser observada através do LGR (Lugar Geométrico das Raízes) visto pela função *sisotool* do MatLab, conforme Figura 3.8.

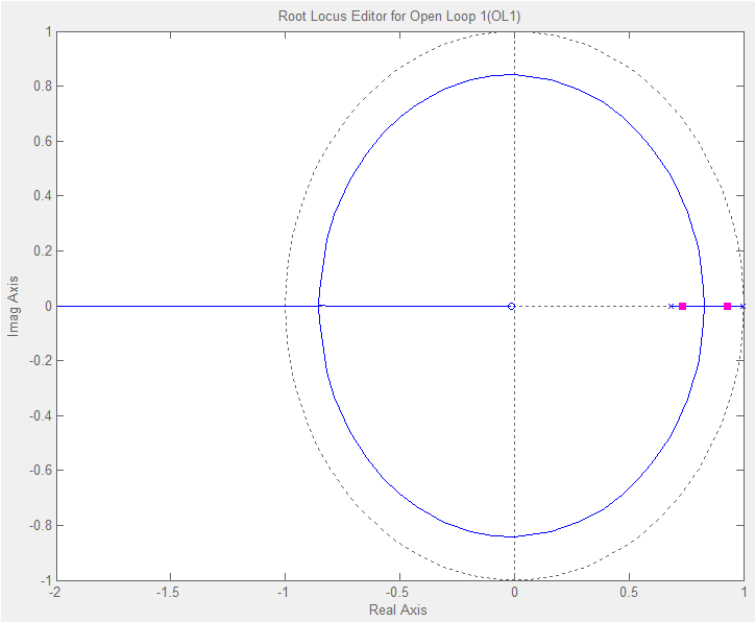

Figura 3. 8 – LGR da altura sem controlador.

A função de transferência em malha aberta Z_a possui um zero em 0.013 e outro no infinito negativo. A mesma possui também um polo em 0.685 e outro em 1 (integrador, responsável pelo erro nulo em regime permanente). O projeto de controladores para diminuir o tempo de resposta do sistema através da abordagem com polos e zeros será realizado na seção 5.3 do capítulo 5.

3.5.2 FUNÇÃO ÂNGULO DE ARFAGEM DO QUADRIRROTOR

Conforme já mencionado na seção 3.2 a simetria do AR.Drone resulta em uma mesma dinâmica para o movimento de arfagem (*pitch*) e rolamento (*roll*), tanto é que a identificação do sistema apresentado no kit de desenvolvimento do *SIMULINK*® resultou praticamente em uma mesma função de transferência.

Para apresentar um modelo linear do movimento de arfagem do AR.Drone coube determinar o limite no comando pelo qual o quadrirrotor responde linearmente. Conforme apresentado em [3] o AR.Drone possui ângulo máximo de 8º para arfagem e rolamento, então, decidiu-se trabalhar com esse limite para as simulações e projeto de controladores.

A função de transferência encontrado no modelo do *SIMULINK*®, com tempo de amostragem de 0.065s, é mostrada na equação (3.31).

$$Z_p = \frac{0.1522z - 0.1341}{z^2 - 1.728z + 0.7721} \tag{3.31}$$

Da mesma forma que a seção anterior, foi realizada uma simulação do comportamento do quadrirrotor mediante um comando de ângulo de arfagem de 8º. O diagrama de blocos da simulação pode ser observado na Figura 3.9.

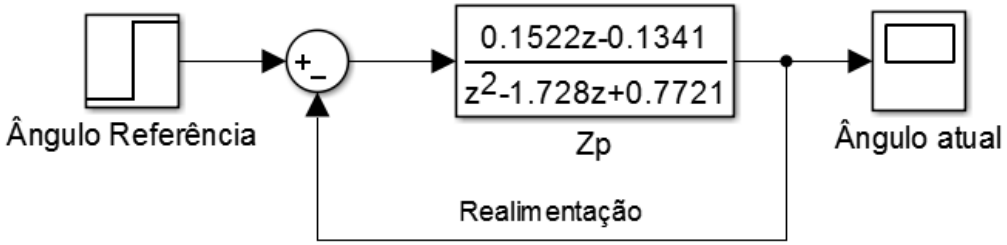

Figura 3. 9 – Diagrama de blocos em malha fechada do ângulo de arfagem sem controlador.

O modelo apresentado por Z_p trabalha com ângulos na métrica de radianos: assim, o ângulo de referência para as simulações é de 0.139 radianos (8º).

O comportamento do sistema sem controlador pode ser observado pela Figura 3.10.

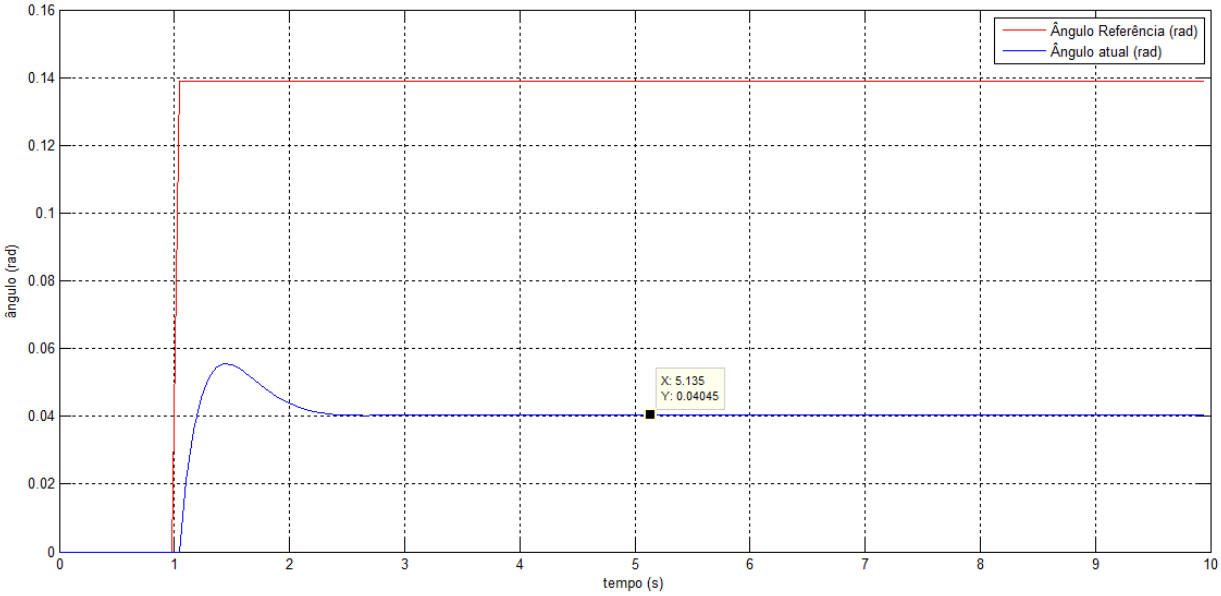

Figura 3. 10 – Resposta do ângulo de arfagem com entrada degrau.

A localização dos zeros e polos pode ser observada na Figura 3.11.

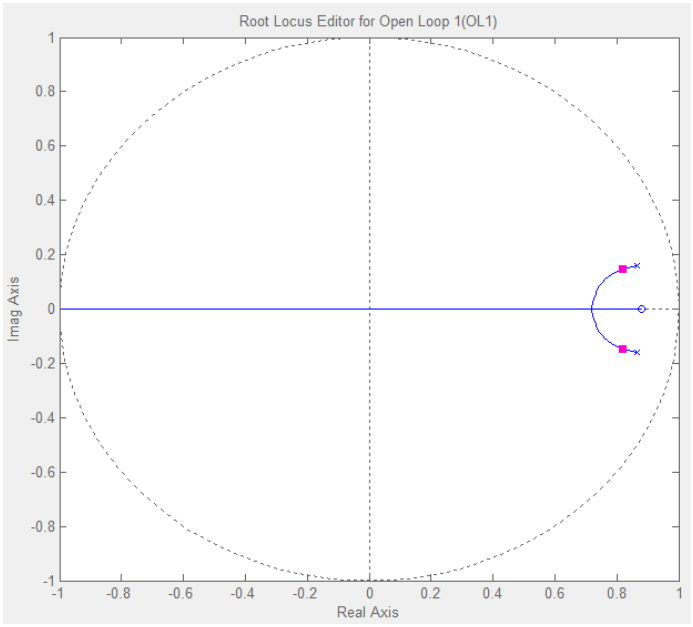

Figura 3. 11 – LGR do ângulo de arfagem sem controlador.

Conforme observamos em Figura 3.10, a resposta do sistema para o movimento de arfagem é de 1.145 segundos (tempo pelo qual a resposta atinge 5% do valor final em regime permanente), com sobressinal de 12.9% e erro em regime permanente de 71.2%. o grande erro em regime permanente é devido à função de transferência ser do tipo 0 (nenhum polo em z=1 em malha aberta), ou seja, erro finito para entrada degrau. O projeto de controladores para o movimento de arfagem pretenderá diminuir o tempo de resposta juntamente com o sobressinal, buscando erro nulo em regime permanente (será apresentado na seção 5.3 do capítulo 5).

CAPÍTULO 4

TÉCNICAS DE PROCESSAMENTO DE IMAGEM E VISÃO COMPUTACIONAL APLICADAS AO PROBLEMA TRATADO

4.1 ASPECTOS GERAIS

A utilização da câmera como sensor no controle de um dispositivo eletrônico requer que haja por trás da aquisição das imagens um processamento rápido, que torne uma simples matriz de valores em dados úteis (neste caso a matriz de pixels). Estes dados, uma vez interpretados da forma correta, devem fornecer as características atuais do sistema (tais como localização, altura, direção e etc.). Para o processamento da imagem existem diversos algoritmos já criados que podem ser implementados em inúmeras linguagens. Neste trabalho serão usados conceitos de aquisição e segmentação de características, para localização de um padrão proveniente das imagens oriundas do AR.Drone. Além disso a segmentação por cor também será abordada. Todo o processamento foi realizado no *software* MatLab, utilizando uma toolbox própria.

Neste capítulo tem por objetivo apresentar conceitos e algoritmos utilizados na elaboração dos códigos de processamento de imagens, os resultados da aplicação desses algoritmos serão mostrados no capítulo 5.

4.2 CARACTERÍSTICAS DE UMA IMAGEM DIGITAL

Uma imagem digital, assim como qualquer imagem, é a representação do que se encontra no ambiente, ou seja, um espaço 3D, em um plano 2D. A captura de uma imagem digital é realizada por dispositivos capazes de discretizar os dados de cor, iluminação ou níveis de cinza e acomodá-los em uma ou várias matrizes, a dimensão dessa(s) matriz(es) depende da aplicação. Cada elemento da matriz, conhecido como ‘pixel’ (abreviação de *Picture element*), possui um valor associado a ele que representa a intensidade de luz refletida naquele ponto. O valor dessa intensidade é dado pelo intervalo de 0 a 255, pois é a capacidade de armazenamento de uma câmera digital com 8 bits, ou seja, de 0 a $2^8 - 1$. [19]

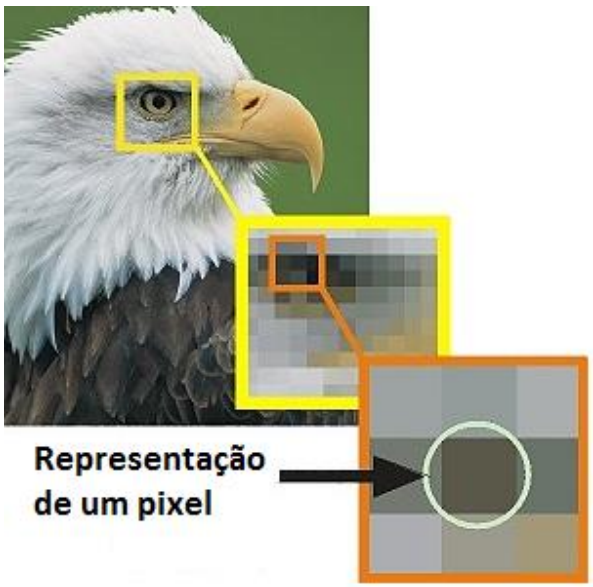

Figura 4. 1 – Exemplo de uma imagem digital[13].

Uma imagem representada em níveis de cinza é caracterizada por possuir apenas uma matriz de observação e cada elemento representa o nível de cinza partindo do mais escuro (valor 0 – preto) ao mais claro (valor 255 – branco) passando pelo cinza puro de valor 127. [19]

Figura 4. 2 – Imagem colorida em (a)[14], imagem em níveis de cinza (b).

Outro tipo de imagem abordada neste trabalho é a imagem do tipo binária, a qual se caracteriza por apresentar apenas dois níveis de cinza, o branco e o preto. Muito utilizada para definir máscaras de convolução de filtros, as imagens binárias são de grande importância no processamento de imagens. Um exemplo de imagem binária é mostrado na Figura 4.3.

[13] http://delbug.ru/elementy_computera/1/8/
[14] http://www.osmais.com

Figura 4. 3 – Imagem binária.

Imagens coloridas possuem, diferentemente da imagem em níveis de cinza, três dimensões, ou seja, três matrizes representando a imagem. O tipo mais comum e muito utilizado em processamento de imagens é o *RGB* [19].

O modelo de representação de imagens coloridas RGB é caracterizado por ser um modelo aditivo de cores, ou seja, com a combinação de três cores, R – *red* (vermelho), G – *green* (verde) e B – *blue* (azul). Neste caso, pode-se criar qualquer cor visualizada pelo olho humano. Uma desvantagem do uso da representação RGB é a não expressividade das cores em relação à iluminação, ou seja, não informação da intensidade de luz apresentada na imagem. A representação *RGB* é utilizada para segmentação por cor, no qual é possível filtrar a cor que for necessária. Neste trabalho o modelo *RGB* é utilizado para separar a cor verde das demais, ou seja, pixel com altos valores na camada G e valores baixos nas demais camadas são interpretados como região a ser segmentada.

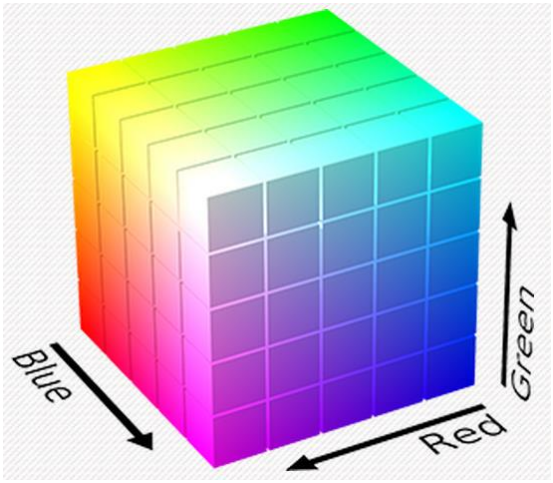

Figura 4. 4 – Cubo do modelo de representação RGB[15].

[15] https://en.wikipedia.org/wiki/HSL_and_HSV

4.3 RELAÇÃO MM/PIXEL DA IMAGEM DIGITAL

Uma relação bastante importante em análise de imagens digitais é a interpretação dos dados da imagem para dados reais, como distância, forma, altura, etc. Neste trabalho, um dado importante fornecido pelo algoritmo de visão computacional é a distância que o centro da imagem do AR.Drone está para o centro do local de pouso. Essa distância é facilmente encontrada em unidades de pixels, mas, para o controle real de trajetória é preciso obter a distância real do quadrirrotor ao alvo em mm. Dessa forma, é necessário obter a relação entre a distância encontrada na imagem (pixels) com a distância real (mm) a cada altura.

Para tanto, foram retiradas quatro fotos da câmera vertical do AR.Drone, cada uma delas a uma distância específica do alvo, através da função *imdistline()* foram encontradas as distâncias em pixels da imagem. Na Figura 4.5 podem ser vistas as imagens obtidas.

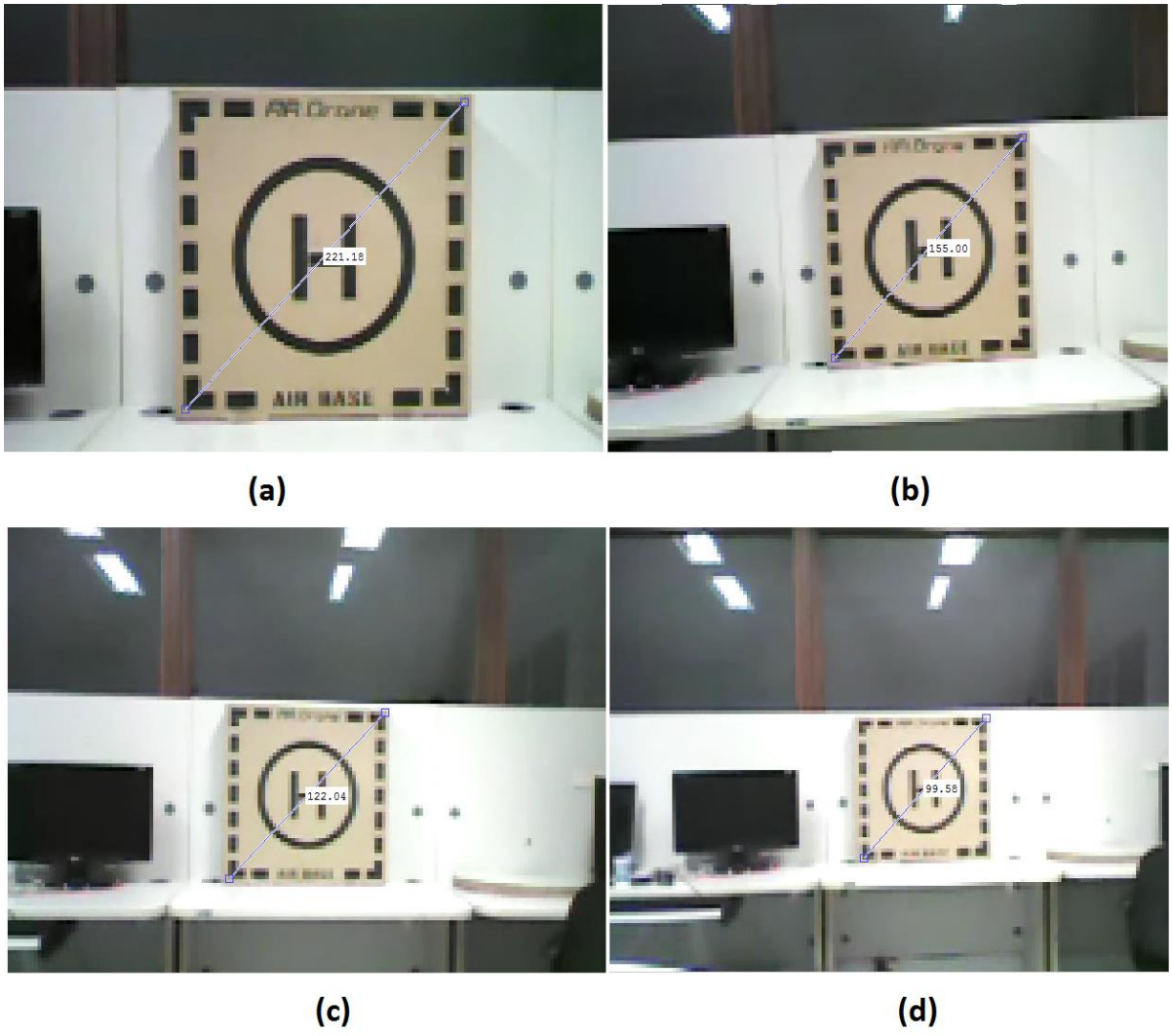

Figura 4. 5 – Imagem distante 1.5 m (a), 2.0 m (b), 2.5 m (c) e 3.0 m (d).

Resumindo as distâncias em pixels, encontradas para a diagonal do local de pouso, e considerando a distância real de 750 mm, são obtidos os dados fornecidos na Tab. 1.

Tabela 1 – Relação mm/pixel da imagem

Distância do alvo (m)	Distância da diagonal (pixels)	Relação mm/pixel
1,5	221,18	3.39
2,0	155,0	4.84
2,5	122,04	6.14
3,0	99,58	7.53

O gráfico da correlação entre a altura do AR.Drone com a relação mm x pixel pode ser visto na Figura 4.6.

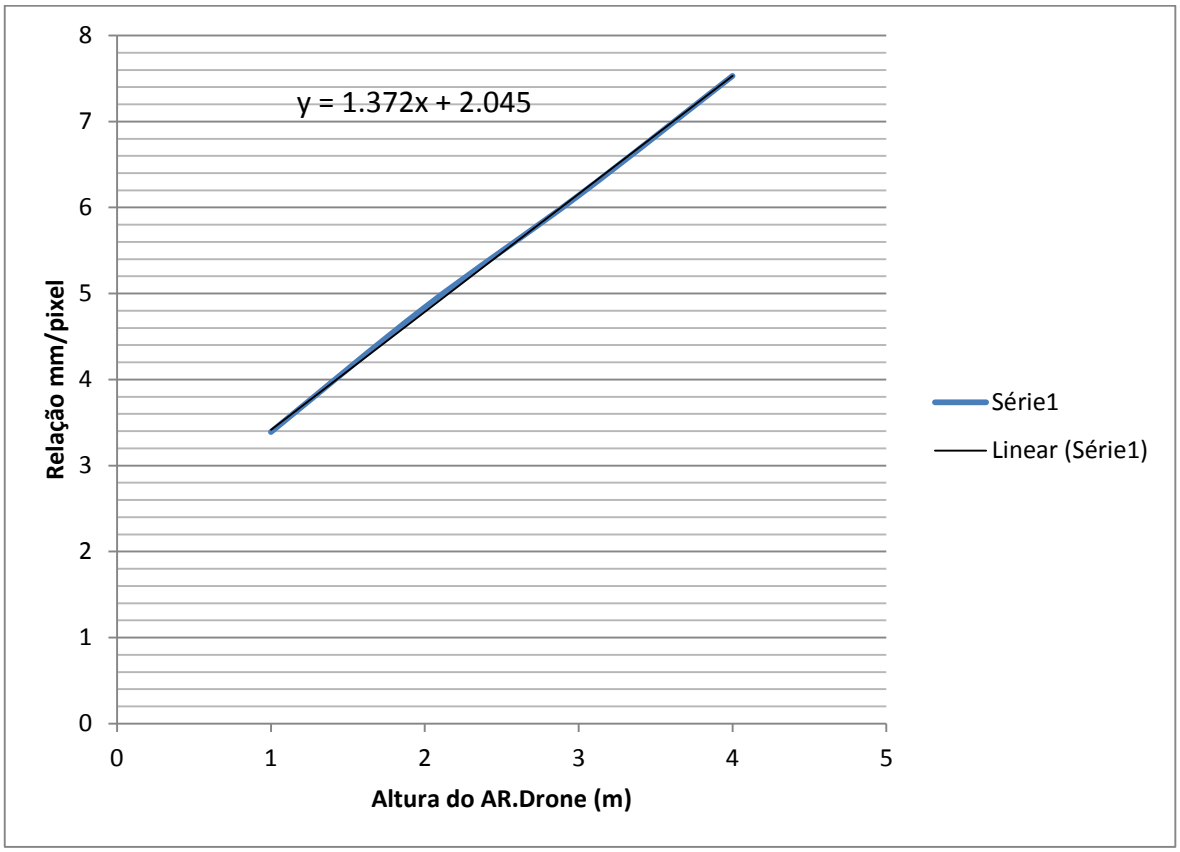

Figura 4. 6 – Gráfico da relação mm/pixel pela altura.

A relação aqui apresentada é limitada para o caso em que o plano da câmera estiver paralelo ao plano da imagem, isso porque a câmera possui distorções nas extremidades da imagem, o qual para uma aplicação com rotação do quadrirrotor ou distâncias maiores do alvo o modelo completo da câmera (calibração) é necessário.

O gráfico da Figura 4.6 pode ser interpretado por uma equação de reta. Dessa forma é possível encontrar a relação mm/pixel para alturas diversas sem que pra isso faça um experimento prévio. Através de interpolação dos pontos obtidos a equação que melhor representa a reta é mostrada na equação (4.1), no qual *Y* é a relação mm/pixel e *X* a altura, realizado também em [3].

$$Y = 1.372 * X + 2.045 \quad (4.1)$$

4.4 MORFOLOGIA

Morfologia é uma área da biologia que estuda a forma e a estrutura de animais e plantas. Aqui usamos a conotação *morfologia matemática* como uma ferramenta para processamento de imagens extraindo informações úteis na representação e descrição da forma de regiões, fronteiras e outros. [1]

A morfologia matemática é aplicada através da linguagem de teoria dos conjuntos. Dessa forma, um conjunto representa uma forma (região) de um objeto em uma imagem. A região descrita é característica por algum atributo comum, como por exemplo, pixels pretos em uma imagem binária. Já descritas na seção anterior, imagens binárias são compostas por conjuntos de espaços bidimensionais de números inteiros Z^2, no qual cada elemento representa as coordenadas (x,y) dos elementos, por exemplo os pixels pretos. [1]

Neste trabalho, alguns conceitos de morfologia matemática foram aplicados em imagens binárias, tais como *dilatação, erosão, abertura* e *fechamento.*

4.4.1 DILATAÇÃO

Antes de tratarmos da aplicação da dilatação algumas definições de teoria dos conjuntos devem ser abordadas, tais como *translação*, *reflexão* e *complemento* de um conjunto [9].

A translação de um conjunto A por um ponto *x* é definido como a soma de cada coordenada do conjunto pelo ponto, em notação matemática temos:

$$(A)x = \{c | c = a + x, a \in A\} \quad (4.2)$$

A reflexão de um conjunto *A* é, resumidamente, uma rotação de *A* em 180º com relação à sua origem, a notação é definida como:

$$\hat{A} = \{c | c = -a, a \in A\} \quad (4.3)$$

O complemento de um conjunto *A* é basicamente todos os pixels que não pertencem a *A* e é definido como:

$$A^c = \{c | c \notin A\} \quad (4.4)$$

A Figura 4.7 ilustra cada uma das definições mostradas acima.

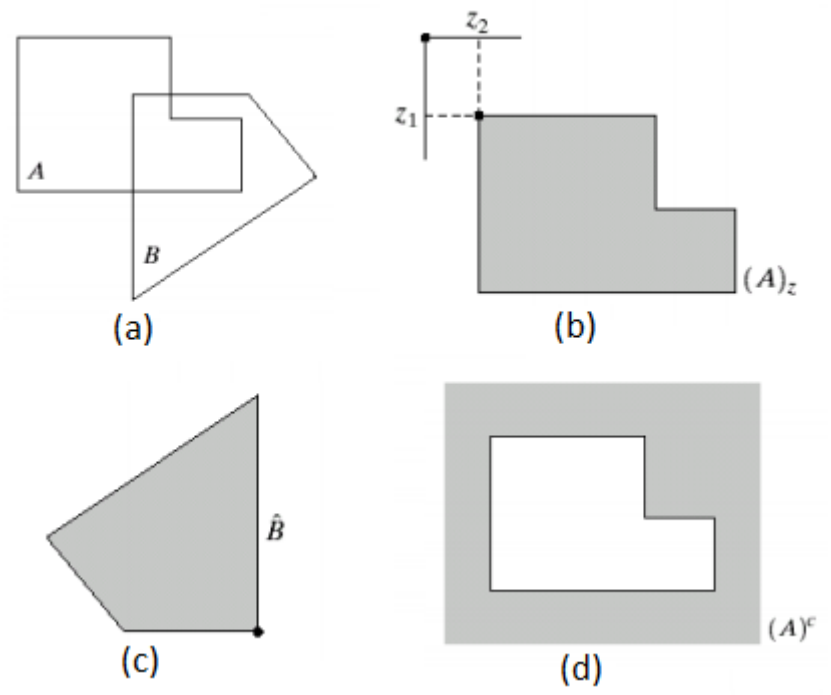

Figura 4. 7 – apresentação de A e B (a), translação de A por z (b), reflexão de B (c) e complemento de A (d), adaptado de [1].

Algumas notações matemáticas podem ser encontradas para definir dilatação. De acordo com [9] é definido como:

$$A \oplus B = \{c | c = a + b, a \in A, b \in B\}, \tag{4.5}$$

onde *A* é o conjunto a ser aplicada a operação de dilatação, *B* é o operador também chamado de elemento estruturante.

Outra definição matemática, apresentada por [1], é:

$$A \oplus B = \{x | (\mathbf{B})x \cap A \neq \emptyset\}, \tag{4.6}$$

onde **B** é a reflexão de B.

A dilatação de *A* por *B* é o conjunto de todos os deslocamentos *x* tais que as regiões de *A* e ***B*** se sobreponham em pelo menos um elemento não nulo [1].

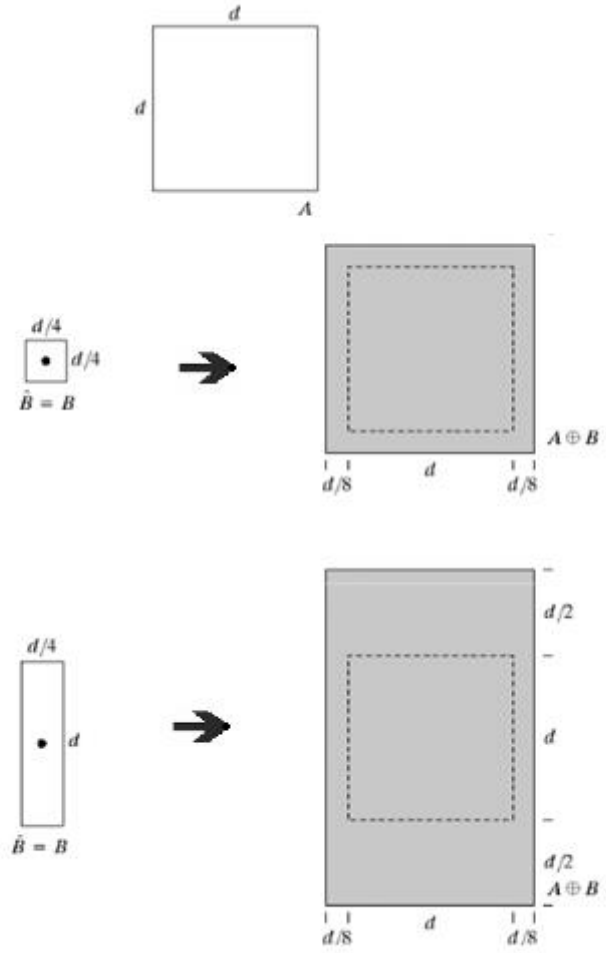

Figura 4. 8 – Dilatação da região A pelo elemento estruturante B [1].

A dilatação é uma forma de expandir uma região adicionando pixels à sua fronteira.

4.4.2 EROSÃO

Outra operação morfológica muito importante em processamento de imagens é a *erosão*. Diferentemente da *dilatação,* a *erosão* faz 'podas' na fronteira de uma imagem. A notação matemática é dada por [9]:

$$A\Theta B = \{c|(B)c \subseteq A\} \tag{4.7}$$

A erosão de *A* por *B* resulta em um conjunto de pontos tais que B transladado esteja contido em *A*.

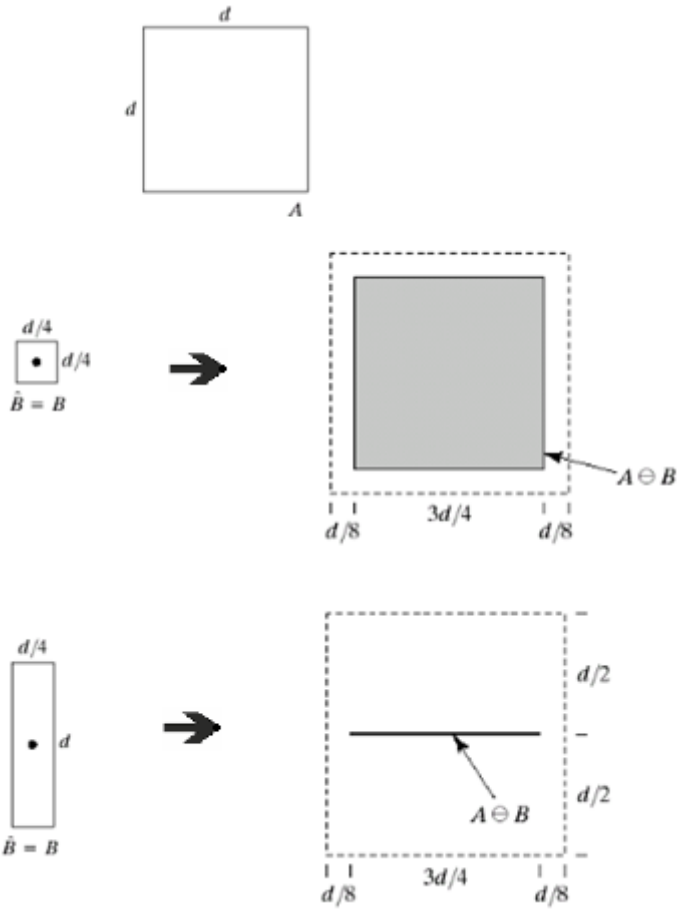

Figura 4. 9 – Erosão da região A pelo elemento estruturante B [1].

A dilatação e a erosão possuem dualidade em suas operações segundo a complementação e reflexão de conjuntos, conforme expressão mostrada na equação (4.8).

$$(\mathrm{A}\Theta\mathrm{B})^{\mathrm{c}} = \mathrm{A}^{\mathrm{c}} \oplus \mathrm{B}^{\mathrm{c}} \tag{4.8}$$

4.4.3 ABERTURA E FECHAMENTO

A partir das operações de dilatação e erosão podem-se usar outros dois tipos de operações, chamados de abertura e fechamento. A operação de abertura é basicamente a aplicação de uma erosão seguido da dilatação [1]. A notação matemática é dada por:

$$\mathrm{A} \circledcirc \mathrm{B} = (\mathrm{A}\Theta\mathrm{B}) \oplus \mathrm{B} \tag{4.9}$$

Abertura é uma operação capaz de eliminar pequenas fendas entre objetos e separar regiões que se encontram por finas linhas, além disso, usa-se abertura para eliminar ruídos espúrios pretos espalhados sobre a imagem, ruídos brancos não podem ser eliminados com essa operação.

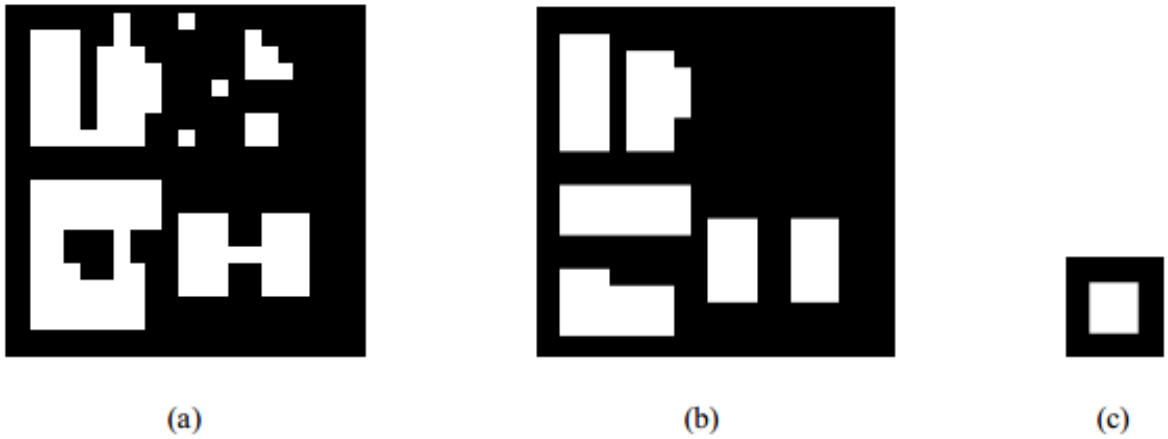

Figura 4. 10 – Imagem de interesse (a), abertura da imagem (b) e elemento estruturante (c)[16].

Fechamento é uma operação similar à abertura, difere-se apenas que o fechamento realiza a dilatação anteriormente à erosão. A notação matemática do fechamento é dada por:

$$A \odot B = (A \oplus B) \Theta B \tag{4.10}$$

Em oposição à abertura, o fechamento é capaz de eliminar ruídos brancos espalhados sobre o objeto, mas para ruídos pretos não há efeito.

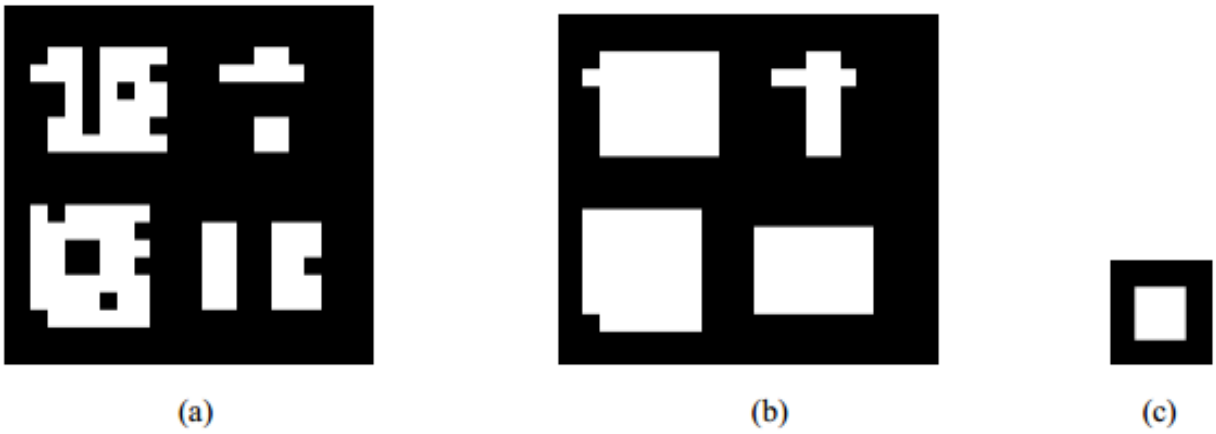

Figura 4. 11 – Imagem de interesse (a), fechamento da imagem (b) e elemento estruturante (c).

Assim como a dilatação e a erosão, as operações de abertura e fechamento possuem dualidade segundo a complementação e reflexão de conjuntos, conforme expressão mostrada na equação (4.11).

$$(A \odot B)^c = (A^c \circledcirc B^c) \tag{4.11}$$

[16] http://laplace.dcc.ufmg.br/npdi/uploads/96a40bea-e420-88f1.pdf

No presente trabalho a utilização da abertura e fechamento foi de fundamental importância para obtenção dos resultados de segmentação por cor, onde é necessário definir de forma precisa uma região de pixels brancos que estão espalhados pela imagem.

4.5 EXTRAÇÃO E DESCRIÇÃO DE CARACTERÍSTICAS

O crescente desenvolvimento dos algoritmos em visão computacional gerou diversas aplicações com grande relevância para a sociedade. Sem dúvida, operações de risco para os seres humanos realizadas por robôs que utilizam câmeras como sensores são indispensáveis atualmente. Porém, mesmo com tantos estudos na área de processamento de imagens e visão computacional, algumas tarefas realizadas facilmente por nós são de grande complexidade computacional.

Diversos algoritmos já foram desenvolvidos, tornando possíveis aplicações bastante interessantes como rastreamento de pessoas, carros e outros objetos com características bem definidas. O reconhecimento de padrões consiste inicialmente na extração de características da imagem a serem confrontadas em outras imagens, através de descritores capazes de encontrar suas semelhanças e realizarem a detecção.

Existem algoritmos tanto para extração (SIFT, SURF) quanto para descrição (SIFT, SURF e HARRIS) de características. Neste trabalho foram utilizados os algoritmos de extração e descrição SURF – Speed-Up Robust Features e o KLT (Kanade-Lucas-Tomasi) para rastreamento, o que tornou possível a comparação entre os dois métodos.

4.5.1 SURF (SPEED-UP ROBUST FEATURES)

SURF (Speed-Up Robust Features) foi proposto originalmente no ECCV realizado na Áustria [4]. SURF é um algoritmo inspirado no SIFT – Scale Invariant Feature Transform – capaz de fornecer características da imagem invariantes à escala, à rotação, porém, mais robusto e computacionalmente mais rápido.

O algoritmo consiste na detecção e extração das características e construção de descritores que representam cada característica. Conforme apresentado no trabalho original [4] há diversas etapas que são realizadas para extrair e descrever as características da imagem.

Na extração de características o algoritmo SURF utiliza inicialmente um detector Hessiano, que é a análise do determinante da matriz Hessiana (equação (4.12)), que é a matriz de derivadas parciais de segunda ordem no ponto de interesse (x).

$$\mathcal{H}(\mathrm{x}, \sigma) = \begin{bmatrix} Lxx(x) & Lxy(x) \\ Lxy(x) & Lyy(x) \end{bmatrix} \tag{4.12}$$

A matriz Hessiana é usada para localizar regiões com ocorrência de variações bidimensionais da textura na imagem, ou seja, em qualquer direção no plano da imagem. Lxx(x) é a derivada parcial de segunda ordem realizada na posição **x** da imagem na direção da coordenada X, da mesma forma Lyy(x) é a derivada parcial de segunda ordem na direção da coordenada Y e Lxy(x) é a derivada parcial de segunda ordem na diagonal da imagem. Para cada uma das derivadas são utilizados filtros Laplacianos. Na Figura 4.12 é possível observar os filtros utilizados nas direções Y e diagonal.

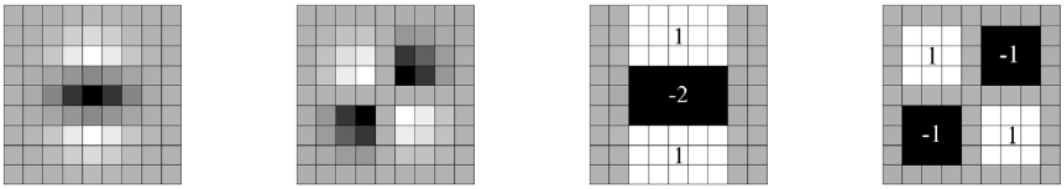

Figura 4. 12 – Primeiro e terceiro filtros: direção Y; Segundo e quarto filtros: diagonal [4].

Observe que nas duas primeiras imagens da Figura 4.12 os filtros possuem valores flutuantes, com intensidades graduais do preto ao branco. No algoritmo SURF os filtros utilizados são obtidos através de aproximação por inteiros dos filtros graduais (terceira e quarta imagens da Figura 4.12), denominados de filtros caixa.

Uma característica do SURF que o torna mais eficiente é a utilização do conceito de imagens integrais no qual se pode acelerar a aplicação dos filtros na imagem. Imagens integrais são capazes de nos fornecer a soma dos valores dos pixels de uma região de qualquer tamanho apenas com duas somas e duas subtrações dos valores dos quatro vértices da região. Cada pixel da imagem integral representa a soma de todos os pixels do retângulo formado acima e à esquerda na imagem original, por exemplo, o pixel na posição D na Figura 4.13 possui o valor da soma dos pixels do retângulo formado pelos vértices opostos O e D na imagem original.

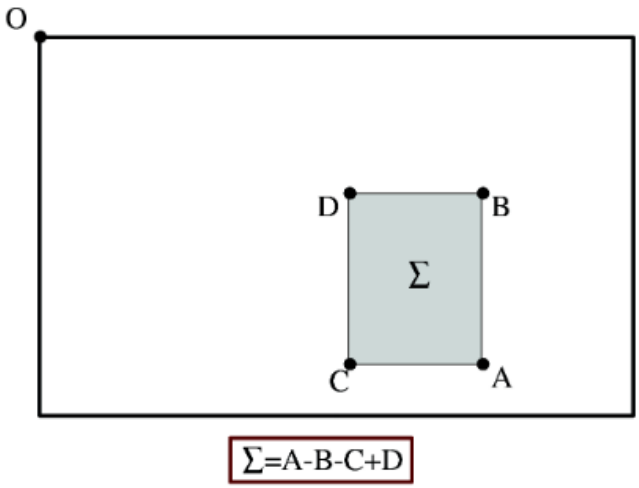

Figura 4. 13 – Definição de uma imagem integral [4].

Dessa forma, para obter a soma dos pixels da região delimitada pelos vértices ABCD na Figura 4.13 basta realizar a seguinte operação, A – B – C + D. Quanto maior é a região analisada mais eficiente será o algoritmo utilizando imagens integrais.

A próxima etapa é calcular o determinante da matriz Hessiana (vide equação (4.12)), o que resulta em um valor numérico. Os filtros caixa da Gaussiana (Figura 4.13) são aplicados em diferentes escalas (chamada pirâmide de escalas, e que será abordado mais a frente). Devido a isso existe uma compensação nos valores de *Lxy,* denominado como *w.* Valores altos do determinante indicam regiões com grande variação em termos de características e valores baixos do determinante indicam que a região analisada é homogênea, ou seja, não possui variações significantes de suas características e atributos.

$$\det(\mathcal{H}\text{aprox}) = \text{LxxLyy} - (\text{wLxy})^2 \tag{4.13}$$

A fim de extrair características da imagem de diversos tamanhos e escalas, outro conceito utilizado no SURF é a pirâmide de escalas. Esse conceito é bastante importante para reconhecimento de padrões, pois nem sempre a imagem alvo está na mesma escala que sua imagem padrão. Neste caso, a pirâmide de escalas é a diminuição na escala da imagem e para cada escala estabelecida aplicar filtros caixa também de diferentes escalas, conforme ilustrado na Figura 4.14.

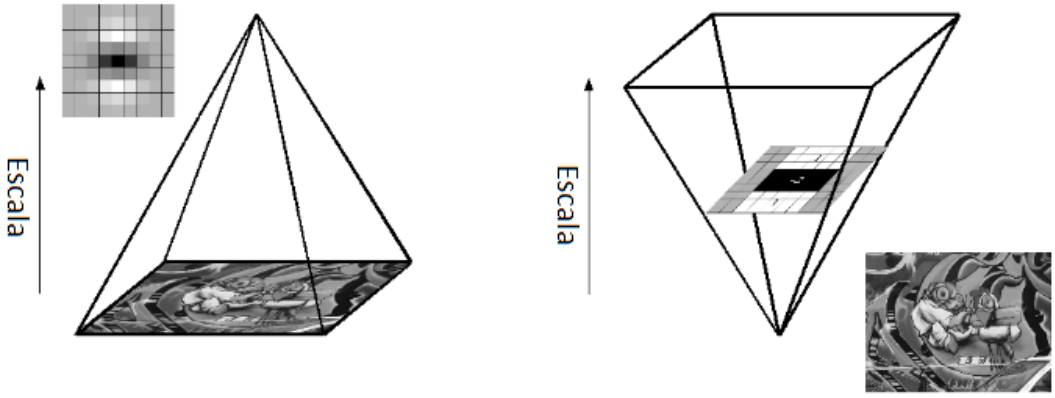

Figura 4. 14 – Pirâmide de escalas, primeira: varia a escala da imagem; segunda: varia a escala do filtro [4].

A princípio a escala dos filtros é multiplicada por dois a cada análise, porém, entre uma escala e outra existem características que passam despercebidas pelos filtros, dessa forma, criam-se escalas intermediárias para extrair o maior número de características possíveis. Daí tem-se o conceito de oitavas (Figura 4.15), tamanhos pelos quais os filtros possuem em cada escala aplicada.

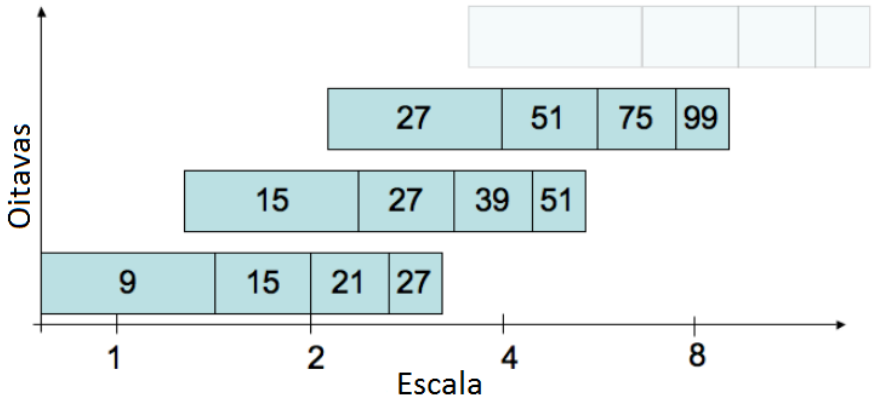

Figura 4. 15 – Definição de Oitavas pela escala [4].

Para a primeira escala tem-se o filtro de tamanho 9, ou seja, 9x9 pixels e os valores vão aumentando conforme à escala. Pode-se reparar que o tamanho dos filtros possui apenas valores ímpares justamente para ter um pixel central de análise (x, do Laplaciano da equação (4.12)).

A aplicação dos filtros resulta em uma matriz cujos valores são os resultados das convoluções nas diferentes escalas. O que se espera é obter valores altos na passagem dos filtros. Porém, o que é exatamente um valor alto para que uma região seja considerada ponto de interesse? Essa pergunta é respondida analisando os valores das matrizes de resultado e seus vizinhos, dessa forma, extrai-se o ponto de interesse cujo valor é maior que seus vizinhos e esses são descartados, essa etapa é chamada de supressão de não máximos em três dimensões. Assim, o ponto de interesse é então interpolado com sua escala e apresentado na imagem conforme método proposto por [12].

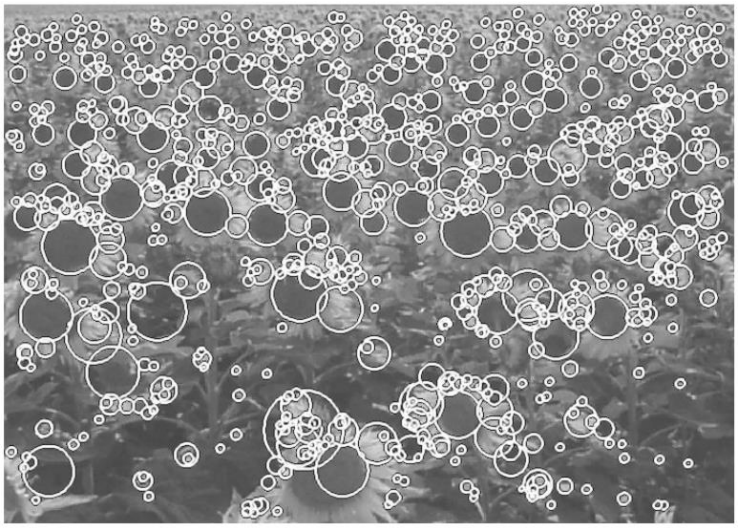

Figura 4. 16 – Exemplo de detecção de características [4].

Conforme mostrado na Figura 4.16 o algoritmo SURF detecta características independentes das escalas em que elas se apresentam, o tamanho dos círculos em torno dos pontos de interesse refere-se à escala em que eles foram detectados.

A extração das características fornece apenas a localização (x,y) da característica e a escala pelo qual ela foi encontrada, informações insuficientes para ter relação entre outras características. Após a detecção das características da imagem vem a descrição dessas características, que é a forma de achar semelhanças entres duas características em imagens distintas.

A descrição da característica extraída nada mais é que gerar um vetor de atributos próprios dela, capaz de reter a informação de escala e orientação, daí vem a capacidade do SURF ser invariante de rotação. Para isso, através das escalas de cada característica, o SURF realiza diversas convoluções nos pixels vizinhos ao ponto de interesse com dois filtros do tipo Haar Wavelets mostrado na Figura 4.17.

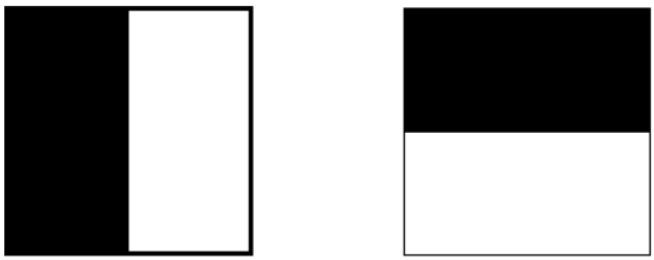

Figura 4. 17 – Filtros nos eixos X e Y de Haar [4].

A próxima etapa é analisar a região centrada no ponto de interesse através de um quadrado de lado igual ao diâmetro da escala e alinhado conforme a orientação principal calculado anteriormente. O quadrado obtido é dividido em sub-regiões com lados do tamanho de ¼ do lado da região principal, gerando assim 16 sub-regiões, conforme mostrado na Figura 4.18.

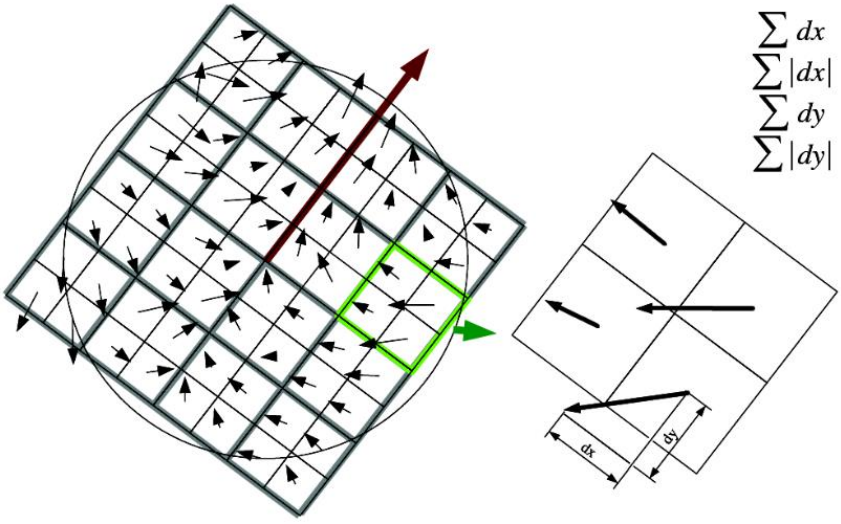

Figura 4. 18 – Regiões e sub regiões, suas orientações e obtenção da somatória [4].

Cada sub-região é então dividida em outras quatro sub-regiões e aplicam-se a elas os mesmos filtros do tipo Haar Wavelets aplicados anteriormente, resultando na orientação nas direções *x* e *y,* dessa vez, em relação à orientação principal. Os valores (dx e dy) dessas quatro sub-regiões são então somados, gerando o seguinte grupo de dados, ∑dx, ∑dy, ∑|dx| e ∑|dy|, no qual os dois primeiros dão ênfase à orientação somando valores que forem negativos e os dois últimos dão ênfase ao comprimento dessas orientações.

Cada sub-região principal possui então quatro valores característicos, gerando no total 16x4 = 64 valores representativos da imagem. O descritor SURF é então um conjunto de vetores (um conjunto para cada ponto de interesse) de atributos cada um com 64 valores.

4.5.2 ALGORITMO KLT (KANADE-LUCAS-TOMASI) DE RASTREAMENTO POR PONTOS DE CARACTERÍSTICAS

O algoritmo KLT de rastreamento é muito usado em tarefas de alinhamento de imagens, onde é conhecida uma subimagem (ou *template*) *T(x)* que se deseja rastrear em uma imagem *I(x)* [10]. Esse rastreamento é realizado através do método de soma dos quadrados da diferença de intensidade dos pixels proposto por Lucas e Kanade em 1981 [5], juntamente com a análise dos pontos de características comuns entre as duas imagens. A escolha dos melhores pontos de características a serem rastreados é apresentada por Tomasi e Kanade em 1991 [8].

A detecção dos pontos de características na subimagem é realizada através do detector de *Harris* [6] que encontra mudanças na textura tanto nas direções *x* e *y* quanto nas diagonais.

As etapas aqui apresentadas do algoritmo fazem parte de uma generalização para modelos paramétricos de movimento em translação. Diversos outros tipos de movimentos podem ser encontrados como a rotação, mas, neste trabalho o estudo da translação foi suficiente [10].

Alguns conceitos devem ser abordados antes de apresentar as etapas do algoritmo. Um deles é o conceito de translação de uma imagem realizada através de uma função vetor de valores, denominada *W(x;p)* conforme equação (4.14).

$$\mathrm{W}(\mathrm{x};\mathrm{p}) = (x + p1, y + p2) \tag{4.14}$$

A translação é um movimento com 2 graus de liberdade e o Jacobiano da função vetor de valores é a matriz identidade, conforme mostrado na equação (4.15).

$$\frac{\partial \mathrm{W}}{\partial \mathrm{p}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.15}$$

O detector de Harris é um dos mais conhecidos detectores de textura na área de processamento de imagens e sua abordagem aqui mostrada foi extraída de [8] conforme a equação (4.16):

$$\mathrm{DH} = \begin{bmatrix} \sum \mathrm{Ix}^2 & \sum \mathrm{IxIy} \\ \sum \mathrm{IyIx} & \sum \mathrm{Iy}^2 \end{bmatrix}, \tag{4.16}$$

onde Ix e Iy é a variação na intensidade dos pixels na imagem nas direções *x* e *y*, a combinação das duas ($IxIy$ e $IyIx$) é a variação na intensidade dos pixels nas duas direções diagonais.

A partir dos conceitos apresentados pode-se mostrar a equação geral do algoritmo KLT para o movimento de translação cuja prova utiliza a Soma dos Quadrados da Diferença de Intensidades que não será abordada neste trabalho:

$$\Delta \mathrm{p} = \mathrm{H}^{-1} * 2 * \sum_x \left[\nabla I \left(\frac{\partial W}{\partial p}\right)\right]^T \Big(T(x) - I\big(W(x;p)\big)\Big), \tag{4.17}$$

onde $\Delta \mathrm{p}$ é o deslocamento do *template* na imagem, H^{-1} é o detector de Harris, ∇I é o gradiente da imagem, $\frac{\partial W}{\partial p}$ é o Jacobiano da função de vetor de valores da translação, $T(x)$ é o template a ser rastreado e $I(W(x;p))$ é o percorrer na imagem através do vetor de valores de deslocamento (nada mais é que o recorte da imagem em diversos pontos de análise). Dessa forma, para calcular $\Delta \mathrm{p}$ seguem as etapas do algoritmo:

1. Recortar a imagem conforme o vetor de valores, obter $I(W(x;p))$;
2. calcular o erro das imagens recortadas, $T(x) - I(W(x;p))$;
3. calcular o gradiente da imagem, ∇I;
4. resolver o Jacobiano para a função vetor de valores, $\frac{\partial W}{\partial p}$;
5. calcular $\nabla \mathrm{I}\left(\frac{\partial \mathrm{W}}{\partial \mathrm{p}}\right)$;
6. calcular a inversa da matriz H que nada mais é do que o detector de Harris, H^{-1};
7. multiplicar o resultado dos itens 5 e 2, $\left[\nabla \mathrm{I}\left(\frac{\partial \mathrm{W}}{\partial \mathrm{p}}\right)\right]^{\mathrm{T}} (\mathrm{T(x)} - \mathrm{I(W(x;p))})$;
8. calcular $\Delta \mathrm{p}$;
9. para o menor valor encontrado de $\Delta \mathrm{p}$ atualizar a posição do template na imagem, $\mathrm{p} \rightarrow \mathrm{p} + \Delta \mathrm{p}$;

Em [4], foi proposta uma abordagem diferente desse algoritmo buscando assim otimizá-lo. A diferença é apenas no cálculo do gradiente do *template* $T(x)$ ao invés da imagem. Por questões óbvias o *template* possui menor dimensão em relação à imagem

total, o que torna mais rápido o cálculo do gradiente. Para ilustrar esse algoritmo na Figura 4.19 mostra o fluxo das etapas do algoritmo:

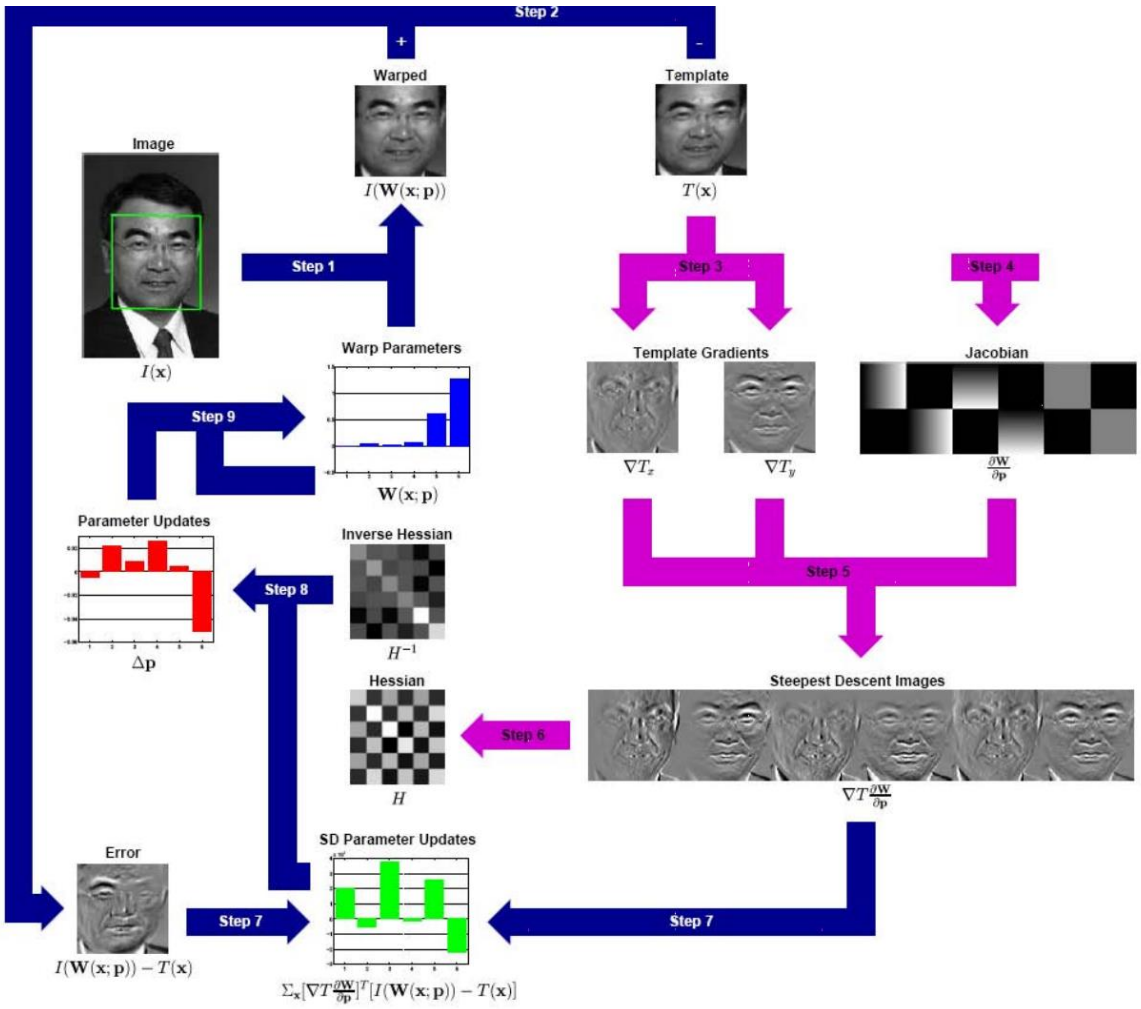

Figura 4. 19 – Etapas do algoritmo KLT [10].

Apesar do proposto por [4], o algoritmo utilizado representa as etapas enumeradas anteriormente.

CAPÍTULO 5

RESULTADOS

5.1 ASPECTOS GERAIS

Este capítulo visa apresentar os resultados obtidos durante todo o processo de desenvolvimento do trabalho. Serão apresentados os diversos códigos implementados com análise comparativa entre eles. As respostas dos sistemas controlados são obtidas via simulação e mostradas também neste capítulo.

As diferentes etapas desenvolvidas neste projeto seguem o fluxo de informação mostrado na Figura 5.1.

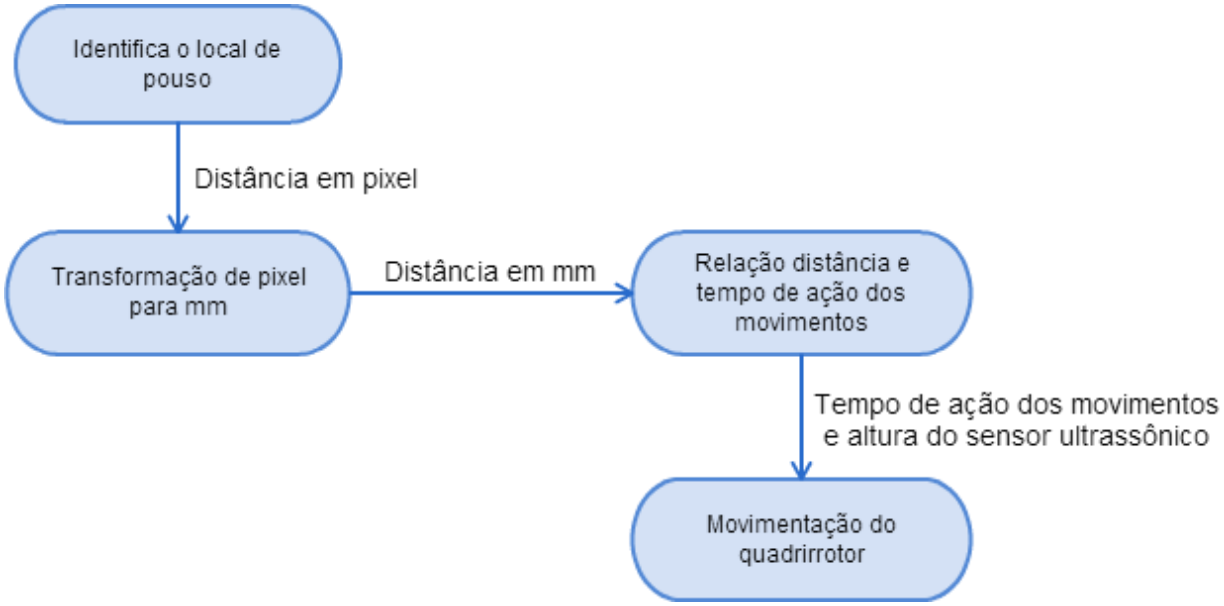

FIGURA 5. 1 – Diagrama do fluxo das informações da plataforma desenvolvida

A transformação de pixel para mm é aquela realizada pela interpolação da Figura 4.6 e a relação distância e tempo de ação dos movimentos será mostrada na Seção 5.4.

5.2 ALGORITMOS EM VISÃO COMPUTACIONAL

Todo o processo de elaboração dos algoritmos baseou-se no reconhecimento de um local de pouso com imagem pré-definida. Desta maneira, é em cima da imagem desse local que todo o trabalho foi desenvolvido, conforme mostrado na Figura 5.2.

Figura 5. 2 – Imagem do local de pouso.

No início dos estudos para criação dos algoritmos deparou-se com o aspecto de iluminação do ambiente. Desejava-se criar um algoritmo de reconhecimento por segmentação por cor, emitida por quatro *leds* encontrados nos vértices do local de pouso (Figura 5.2). Porém, testes iniciais constataram que a câmera do AR.Drone não era capaz de distinguir a cor em ambientes muito claros (com incidência solar abundante), devido à sua baixa resolução na captura das imagens (o tamanho da imagem proveniente da câmera abaixo do AR.Drone é de 176 x 144 pixels). Também devido à resolução não foi possível utilizar a imagem do local de pouso para identificação em ambiente escuro, pois a clareza dos traços da imagem é de fundamental importância para extrair as principais características da mesma.

Devido a esse aspecto, foi então elaborado um projeto que consistia na criação de códigos independentes. O primeiro para ambientes escuros, baseado na segmentação por cor, conforme a ideia inicial, e outros dois para ambientes claros, baseado no reconhecimento de padrões através de extração e descrição de características com os algoritmos SURF e KLT, tomando como referência a imagem do local de pouso.

5.2.1 ALGORITMO DE SEGMENTAÇÃO POR COR

O algoritmo de segmentação por cor consiste em analisar cada *frame* proveniente da câmera do AR.Drone a fim de se obter a cor emitida pelos *leds*, localizados nas extremidades do local de pouso (Figura 5.3).

Figura 5. 3 – Detalhe de um dos leds localizados no local de pouso.

Cada *frame* é ‘varrido’ pixel a pixel, nas três camadas *RGB*, a procura pela cor padrão dos *leds*. Foram utilizados *leds* de alto brilho na cor verde. O intervalo de valores dos pixels de cada camada foi obtido experimentalmente. Assim, sem que houvesse falsos positivos, os valores que correspondem à identificação correta são: camada R, valores menores que *241;* camada G, valores maiores que *230*; camada B, valores menores que *250.* Os valores foram determinados experimentalmente valendo que o ambiente foi preparado para isso. Sabe-se que em um ambiente não controlado é possível existir diversos falsos positivos, dessa forma, essa solução limita-se ao ambiente de teste considerando que a região de pouso não teria nenhuma interferência por cor proveniente de objetos ou iluminação externa.

Cada posição dos pixels identificados como representativos da cor emitida dos *leds* eram mapeados em uma imagem binária, atribuindo a essas posições cor branca e todas as outras com a cor preta. Essa identificação nem sempre gerava regiões bem definidas de aglomerados de pixels representando a cor, havia diversos pixels brancos espalhados pela vizinhança. Dessa forma, foram utilizadas operações morfológicas para ‘juntar’ esses pixels em uma só região. Aplicou-se então a operação de fechamento com dois tamanhos para o elemento estruturante, um de forma circular de raio *20* pixels para a operação de dilatação e outro também de forma circular de raio *8* pixels para a operação de erosão.

Obtiveram-se assim as regiões identificadas para a emissão de luz de cada *led,* conforme mostrado na Figura 5.4.

Figura 5. 4 – Imagem do AR.Drone (a), Identificação da cor (b), Aplicação de abertura (c) e Aplicação de fechamento (d).

A partir das regiões identificadas foram calculados os centroides de cada uma delas a fim de realizar a demarcação dos vértices do local de pouso, auxiliando também no cálculo do ponto central da imagem utilizada para estabelecer a referência do deslocamento a ser realizado pelo AR.Drone. A identificação e o ponto central são mostrados na Figura 5.5.

Figura 5. 5 – Local de pouso identificado com marcação de seu centro.

Algumas dificuldades foram encontradas ao longo dessa parte do projeto, uma delas foi a não captura da cor de *leds* comuns, devido a baixa qualidade das imagens do AR.Drone. Dessa forma, definiu-se a utilização de *leds* de alto brilho. Outro problema obtido foi em relação à iluminação do ambiente, apesar da luz dos *leds* há um limite mínimo de iluminação do ambiente que a câmera do AR.Drone consegue capturar. Assim, em alguns ambientes testados a imagem capturada não tinha nenhuma cor (preto). Neste caso, outros testes foram feitos a fim de estabelecer a iluminação externa correta para a aplicação.

5.2.2 ALGORITMO DE RECONHECIMENTO SURF

Como mostrado no Capítulo 3 existem outros algoritmos tanto para extração quanto para descrição de características. Os dois principais algoritmos de reconhecimento

implementados utilizaram a técnica de extração e descrição de características do SURF (Speed Up Robust Features) e a técnica de rastreamento KLT (Kanade-Lucas-Tomasi).

Os parâmetros que geraram melhores resultados na identificação foram os seguintes: (a) número de oitavas, 3; (b) número de níveis de escalas, 4; (c) *threshold* para a seleção das características mais fortes, 1000.

Inicialmente, são detectadas as características da imagem padrão do local de pouso através da função do MatLab *detectSURFFeatures()*, mostrado na Figura 5.6.

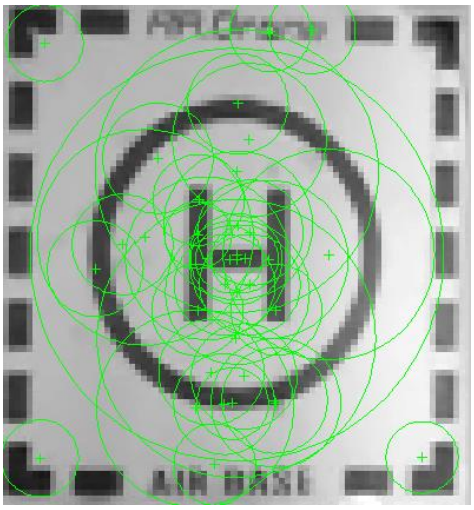

Figura 5. 6 – Características identificadas na imagem padrão do local de pouso.

São detectadas também as características do *frame* a ser analisado proveniente da câmera do AR.Drone. A Figura 5.7 mostra um *frame* capturado.

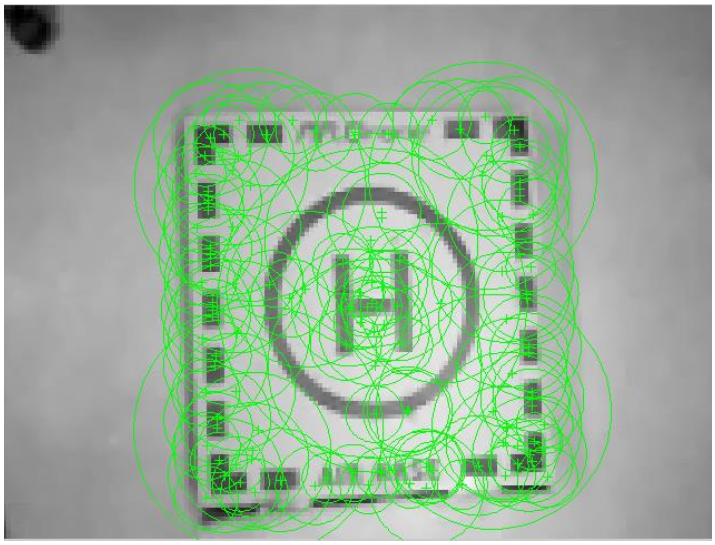

Figura 5. 7 – Identificação das características em imagem proveniente do AR.Drone.

As características das duas imagens são combinadas a fim de se encontrar a correspondência entre elas, aplicando limiar de comparação o resultado, com o algoritmo SURF, pode ser observado na Figura 5.8.

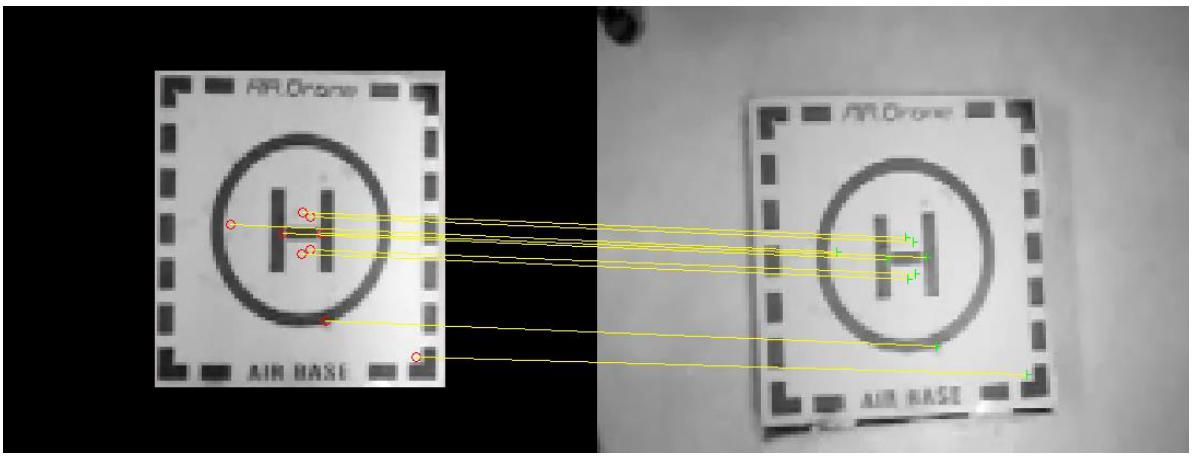

Figura 5. 8 – Correspondência entre as características encontradas nas duas imagens.

A partir da posição do mapeamento das características é feita uma estimativa da escala representativa do local de pouso na imagem do vídeo, Isso é feito pela função *estimateGeometricTransform(),* dessa forma é possível mostrar a região identificada conforme mostrado na Figura 5.9.

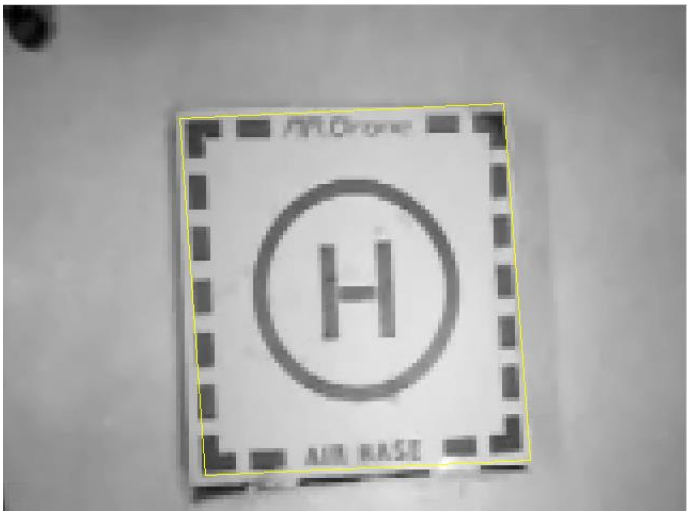

Figura 5. 9 – Identificação do local de pouso.

Conforme o próprio artigo em [4] relata que o SURF é invariante à escala e rotação da imagem. Segue a Figura 5.10 com a identificação do local de pouso mesmo que parcialmente ocluso e inclinado.

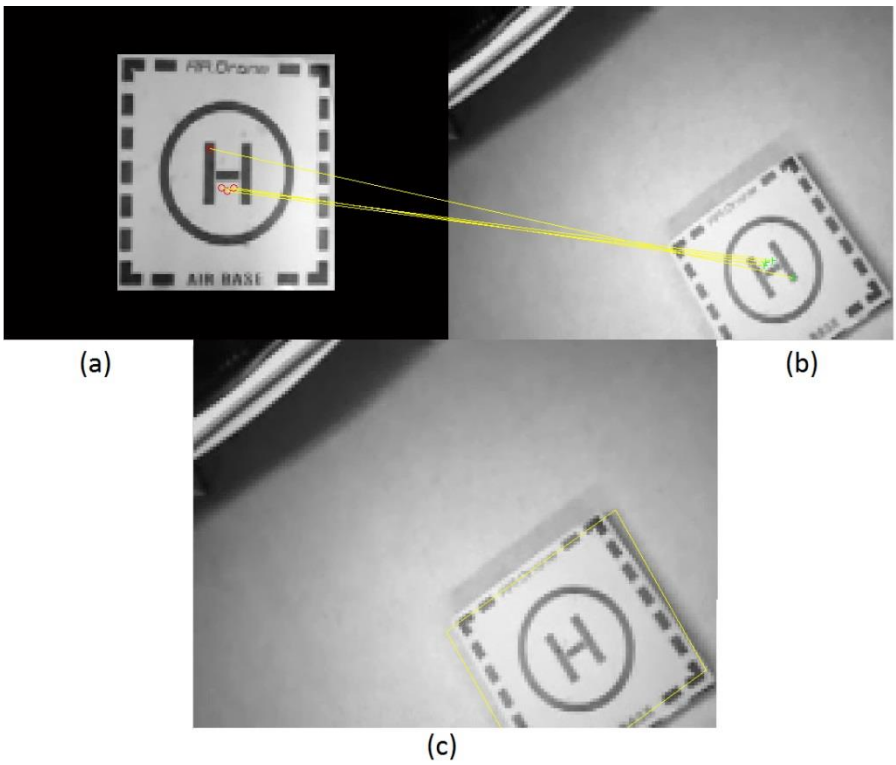

Figura 5. 10 – Exemplo de identificação com deslocamento na imagem, local de pouso padrão (a), *frame* capturado (b) e identificação (c).

Os resultados mostrados nas figuras anteriores foram obtidos aplicando o algoritmo básico com a imagem do local de pouso padrão, que chamaremos de ROI (Região de Interesse – em português) padrão. Porém, ao analisar um vídeo inteiro nem todos os *frames* conseguem ter a identificação com resultados apresentáveis. Dessa forma, foi elaborada uma sequência de processamento na imagem a fim de aumentar a qualidade dos resultados. Essa contribuição, criada especificamente para este trabalho, é representada pelo fluxograma mostrado na Figura 5.11.

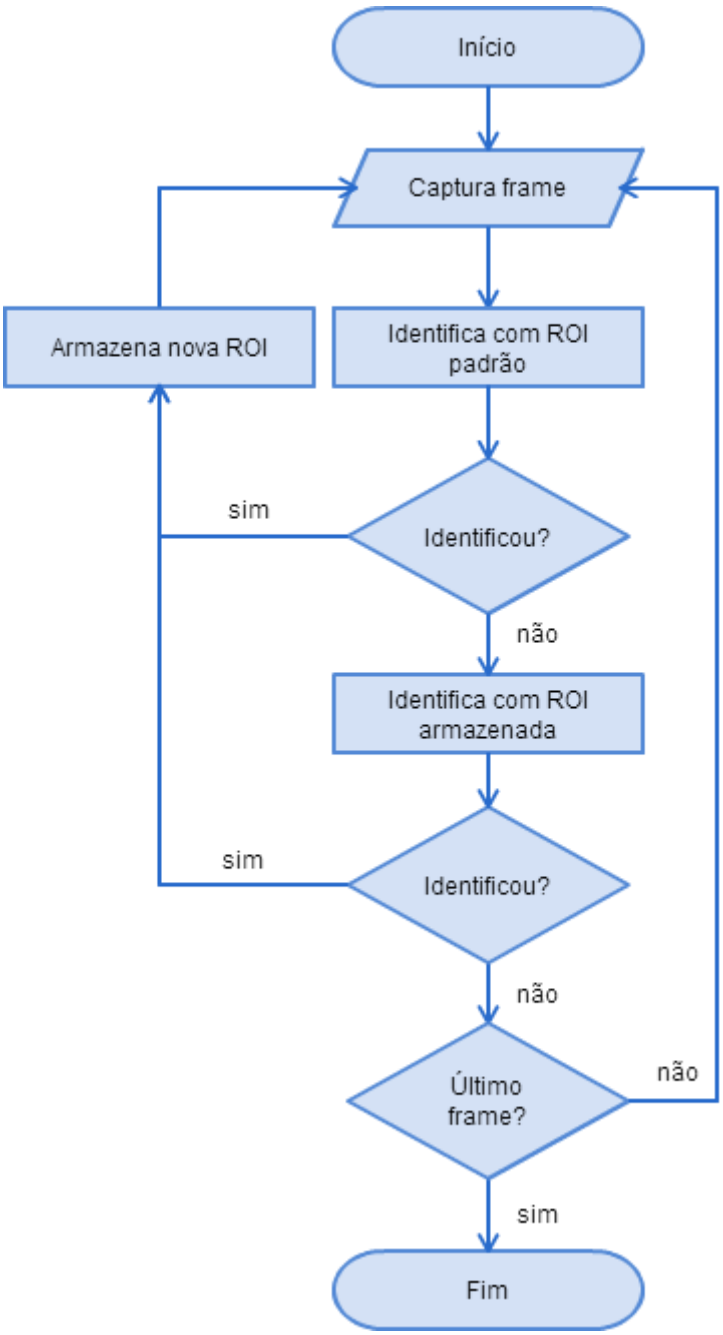

Figura 5. 11 – Fluxograma do algoritmo SURF de identificação.

Diversos vídeos foram feitos a fim de validar o algoritmo proposto, com diferentes intensidades de luz, feitos a partir do voo do AR.Drone e também com a movimentação dele pelas mãos do usuário. Neste caso, a qualidade dos resultados foi dimensionada através da quantidade de *frames* constantes do local de pouso no vídeo e em quais deles o algoritmo foi capaz de identificar. Essa análise é mostrada pela Tab. 2.

Tabela 2 – *Frames* identificados em cada vídeo com SURF

Vídeo	Nº total de *frames* com a imagem do alvo	Nº de *frames* identificados (SURF)	Porcentagem (%)
Vídeo1	201	81	40.3
Vídeo2	187	101	54
Vídeo3	280	120	42.8
Vídeo4	169	93	55
Vídeo5	1371	639	46.6

O vídeo *Vídeo1* foi capturado durante voo do AR.Drone, já os vídeos Vídeo2, Vídeo3, Vídeo4 e Vídeo5 foram capturados com o AR.Drone sendo movimentado pelas mãos do usuário. O que fez com que Vídeo2 e Vídeo4 tivessem resultados melhores foi o fato de o ambiente ter iluminação uniforme e menos intensa que nos vídeos Vídeo3 e Vídeo5.

Observando os *frames* em que o algoritmo não foi capaz de identificar podem-se constar características comuns entre eles, as quais estão com a imagem borrada devido ao rápido movimento do AR.Drone (normalmente em vídeos capturados em voos – Vídeo1) ou outros que pelo pequeno tamanho do local de pouso e baixa resolução da imagem não apresentam formas bem definidas capazes de serem extraídas. A fim de constatar essa última análise, fez-se vídeos através do celular (resolução 36 vezes maior que da câmera do AR.Drone) e os resultados foram consideravelmente mais constantes.

5.2.3 ALGORITMO KLT (KANADE-LUCAS-TOMASI)

O código criado utilizando a técnica KLT de rastreamento foi constituído para abrir outra linha de raciocínio a cerca de como é possível identificar e rastrear um objeto por um vídeo. É um algoritmo não tão complexo (diferentemente do algoritmo SURF). Neste caso, o KLT se mostrou uma ferramenta eficiente quando aplicada em ambientes com certa restrição, como a movimentos do objeto rastreado. Como introduzido no Capítulo 4 o KLT foi utilizado para estimar o movimento translacional do local de pouso.

No início dos testes foi observada a ineficiência do KLT para localizar o local de pouso através da ROI original mostrada na Figura 5.2, devido à escala e dimensão das imagens. Dessa forma, definiu-se a criação de um código que integra o SURF e o KLT, a fim de melhorar os resultados obtidos na Seção 5.2.2.

O algoritmo intitulado de *identifica_dia_KLT_SURF* realiza as seguintes tarefas: (a) usa-se o SURF para identificar o local de pouso e a cada identificação a região encontrada é (b) a imagem é atualizada para o KLT. Resumindo, é necessário que o SURF identifique o

local de pouso a primeira vez para que se obtenha um objeto de rastreamento com o KLT. A contribuição para o esquema geral do algoritmo é mostrado na Figura 5.12.

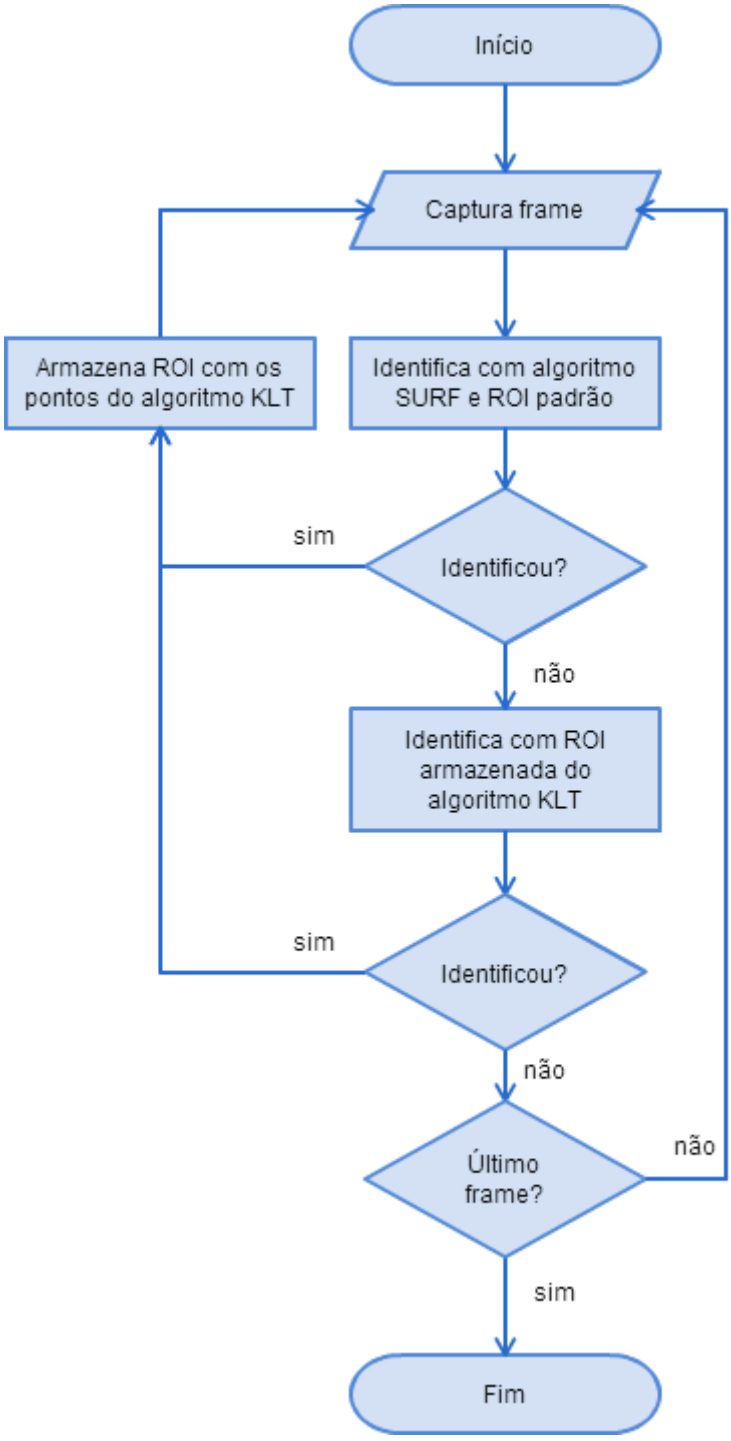

Figura 5. 12 – Fluxograma do algoritmo KLT de identificação.

Conforme mostrado em Figura 5.12, o KLT entra em ação somente quando o SURF não é capaz de identificar o alvo. Assim que o SURF identifica o alvo, são extraídos da região identificada os pontos de características e armazenados (Figura 5.13) junto a um objeto de rastreamento – *tracker,* que faz a procura do deslocamento de translação do alvo no *frame* posterior.

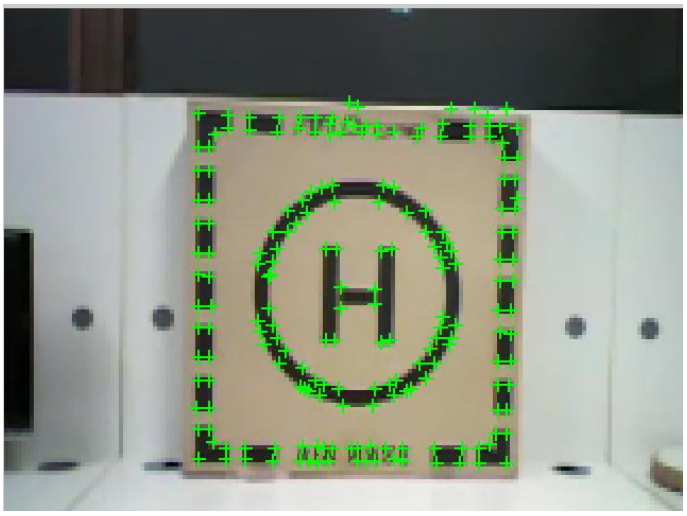

Figura 5. 13 – Características extraídas com o algoritmo KLT.

Os mesmos vídeos analisados na Tab. 2 foram utilizados no código criado para testar a integração do SURF junto ao KLT e o resultado é mostrado na Tab. 3.

Tabela 3 – Resultado da identificação com o KLT

Vídeo	Nº total de *frames* com a imagem do alvo	Nº de *frames* identificados (KLT)	Porcentagem (%)
Vídeo1	201	111	55.2
Vídeo2	187	102	54.5
Vídeo3	280	199	71
Vídeo4	169	142	84
Vídeo5	1371	919	67

Neste caso houve um aumento significativo na quantidade de *frames* identificados com a integração do SURF e do KLT. Os dados de análise finais serão apresentados no item 5.2.4.

5.2.4 ANÁLISE FINAL DO PROCESSAMENTO DE IMAGENS

Os resultados mostrados nos itens 5.2.1, 5.2.2 e 5.2.3 foram considerados como relevantes para este trabalho. A evolução dos conhecimentos do ambiente de projetos em processamento de imagens foi muito perceptiva. Mas, por outro lado, as dificuldades encontradas mostraram-me o quão complexo é lidar com o ambiente real a fim de se criar sistemas que funcionem independentemente do tipo de aplicação.

O algoritmo criado para identificar o local de pouso durante a noite foi o que mostrou os melhores resultados. A segmentação por cor em ambientes controlados é uma tarefa relativamente tranquila. Um dos problemas tratados nesta etapa foi o espalhamento da luz na lente da câmera do AR.Drone, como mostrado na Figura 5.14. Devido ao alto brilho do

led não foi possível controlar a intensidade apenas com a corrente que o alimentava, dessa forma colocou-se um aparato envolto ao *led* para diminuir assim o espalhamento da luz na lente da câmera.

Figura 5. 14 – Espalhamento da luz na lente da câmera do AR.Drone.

Outro problema encontrado, foi que em ambientes extremamente escuros a câmera do AR.Drone não era capaz de capturar a intensidade da luz do *led*. Dessa forma, a luz ambiente foi controlada para que fosse possível realizar a aplicação.

Os algoritmos para ambiente claro baseados em extração e descrição de características demandaram um pouco mais de esforço. A resolução da câmera do AR.Drone não permitiu melhores resultados, mas, mesmo assim, permitiu em média 47.7% da identificação dos *frames* com o algoritmo SURF apenas, com a integração do KLT esse número subiu para 66.3% dos *frames* identificados. Considerando apenas os *frames* identificados independentemente do número total de *frames* do vídeo a melhora na identificação utilizando o KLT em conjunto com o SURF foi de, em média, 33.4%. Sendo que em um vídeo cujo ambiente foi mais bem controlado (luz, movimentação do AR.Drone) a melhora chegou a 69.1%.

Diante desses dados, criou-se um terceiro algoritmo para o ambiente claro, no qual apenas a primeira identificação seria realizada pelo SURF e os demais com o KLT. Dessa vez, o KLT não se mostrou eficiente o bastante. Isso porque é um algoritmo variante à escala e rotação, enquanto o local de pouso movia apenas nos eixos *x* e *y* a identificação era perfeita, mas assim que o alvo saia do campo de visão da câmera e voltava o algoritmo não era mais capaz de reconhecê-lo. O KLT se mostrou limitado à aplicação deste trabalho, mas em conjunto com o SURF foi bastante eficaz gerando ótimos resultados.

Outro estudo realizado em relação aos algoritmos aqui implementados foi o de tempo de execução. O tempo de execução é muitas vezes uma barreira para a aplicação do código em ambiente real. Foi feito análise em *frames* que a identificação ocorreu, a média do tempo

de execução por *frame* em cada método foi: algoritmo SURF, 0.1643 segundos; algoritmo KLT, 0.1471 segundos; algoritmo por segmentação de cor, 0.0277 segundos. É de se esperar que o algoritmo SURF resultasse no maior tempo de execução devido aos seus cálculos na extração e descrição das características. Considerando que a captura do vídeo pela câmera do AR.Drone acontece em uma taxa de 60 *frames*/segundo, temos que cada *frame* é obtido em 0.017 segundo. Dessa forma, em uma aplicação prática dos algoritmos não é possível analisar *frame* a *frame* utilizando o SURF e o KLT, com a segmentação por cor seria possível a cada dois *frames*. Com o SURF a análise deveria ser de dez em dez *frames* e com o KLT de sete em sete.

5.3 PROJETO DE CONTROLADORES

Os controladores são responsáveis por gerar respostas desejadas de sistema que naturalmente não seriam capazes de realizá-las. Os controladores aqui abordados foram projetados para atuarem em cascata com a planta, conforme ilustrado na Figura 5.15.

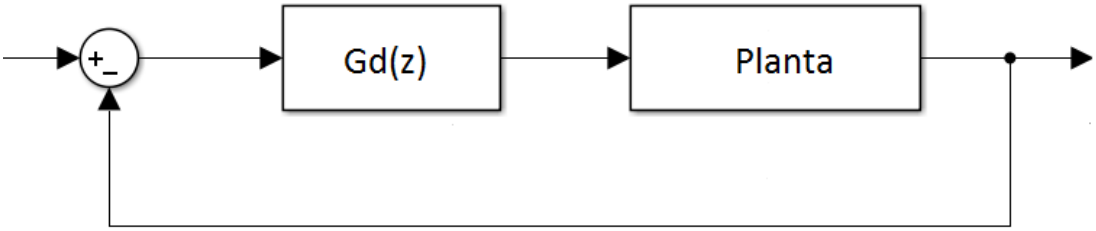

Figura 5. 15 – Diagrama de blocos de controlador em cascata com a planta.

Dois tipos de controladores foram concebidos, um por intermédio do LGR (Lugar Geométrico das Raízes) e outro com resposta *DeadBeat*, para cada sistema apresentado na Seção 3.5.

Controladores por intermédio do LGR são conhecidos por compensadores, pois tem como objetivo compensar com polos e zeros adicionais sem alterar o sistema, de modo que o LGR resultante passe pela localização desejada para o polo para algum valor de ganho [18].

5.3.1 PROJETO CONTROLADOR POR INTERMÉDIO DO LGR PARA A ALTURA

As especificações para a resposta da altura foram definidas como:

$$M_p = 10\,\%\; e\; t_s = 1\; s\,, \tag{5.1}$$

onde M_p é o sobressinal máximo e t_s é o tempo de assentamento da resposta, que são definidos na equação (5.2):

$$M_p = e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}} \; e \; t_s = \frac{4}{\xi w_n}, \tag{5.2}$$

onde ξ é o fator de amortecimento e w_n é a frequência natural de oscilação do sistema. Através da especificação da resposta do sistema obteve-se $\xi = 0.5912$ e $w_n = 6.7664$. Estes parâmetros são utilizados para determinar a localização dos polos do sistema através da equação (5.3).

$$\mathrm{s} = -\xi w_n \pm j w_n \sqrt{1-\xi^2} \tag{5.3}$$

A localização do polo em tempo contínuo é $\mathrm{s} = -4.000 \pm \mathrm{j}5.457$. Para obter o polo em tempo discreto utiliza-se a equação (5.4), onde T é o tempo de amostragem do sistema igual a 0.065.

$$\mathrm{z} = e^{sT} \tag{5.4}$$

Dessa forma, os polos em tempo discreto em malha fechada são $\mathrm{z} = 0.723 \pm \mathrm{j}0.268$. O LGR apresentado pela Figura 3.8 não passa pela localização desejada dos polos, dessa forma, não é possível obter a resposta desejada apenas ajustando o ganho do sistema. O cálculo dos ângulos do zero e dos polos do sistema em relação à um dos polos desejado é feito conforme ilustrado na Figura 5.16.

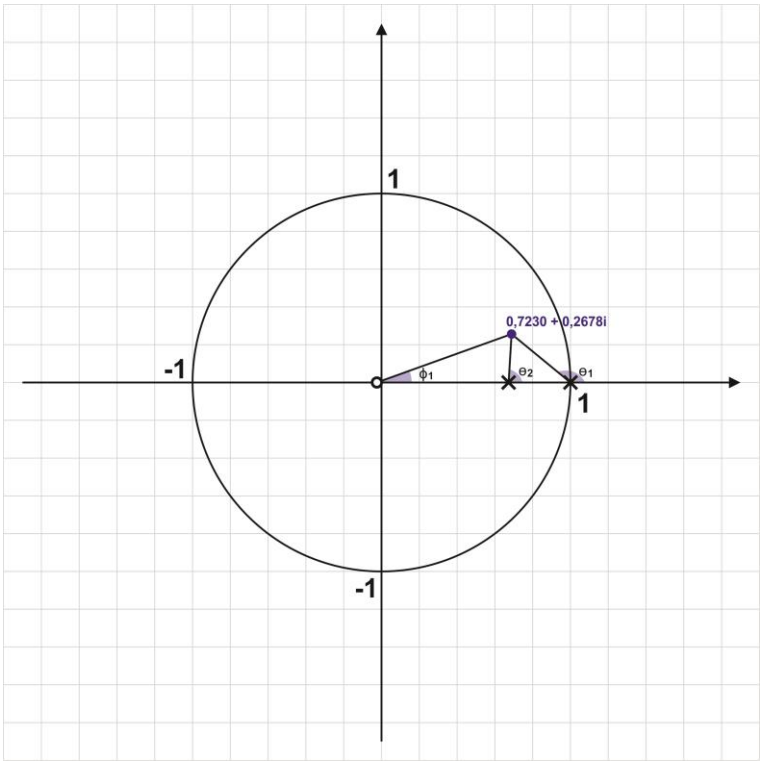

Figura 5. 16 – LGR para cálculo dos ângulos dos polos e zeros da altura.

O valor a ser compensado pelo controlador é de:

$$\theta_c = \sum(\hat{a}ngulos\ polos) - \sum(\hat{a}ngulos\ zeros) - 180º = 17.895º\,. \tag{5.5}$$

Logo o controlador a ser projetado é determinado compensador por avanço de fase devido ao valor positivo do ângulo de compensação. A forma geral do compensador por avanço de fase é definida na equação (5.6),

$$G_d(\mathrm{z}) = \mathrm{K}\frac{\mathrm{z} - z_c}{z - p_c}, onde\ z_c > p_c\,. \tag{5.6}$$

Diversas abordagens para a alocação de zeros e polos de controladores podem ser realizadas, escolheu-se anular um dos polos do sistema através do zero do controlador, dessa forma temos:

$$z_c = 0.685\,. \tag{5.7}$$

A partir daí calcula-se o valor do polo e o ganho K do controlador, no qual obtemos:

$$G_d(\mathrm{z}) = 8.1895\frac{\mathrm{z} - 0.685}{z - 0.5925}. \tag{5.8}$$

Com o controlador projetado foi realizada simulação para verificar se o comportamento da resposta respeitou as especificações desejadas. No *SIMULINK*® foi desenvolvido o diagrama de blocos mostrado na Figura 5.17, no qual a entrada é um degrau de amplitude 1 a partir do primeiro segundo.

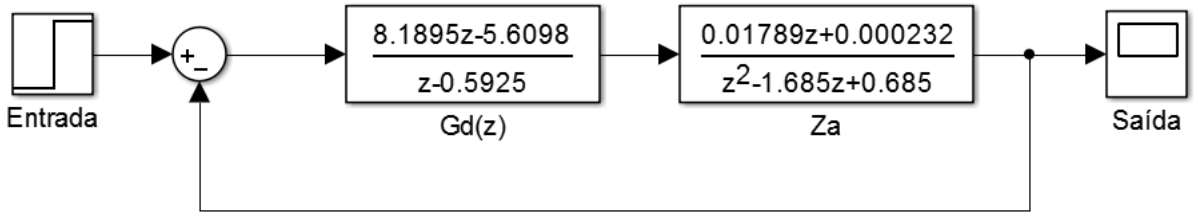

Figura 5. 17 – Diagrama de blocos de simulação do controlador projetado da altura.

A resposta obtida pode ser visualizada pela Figura 5.18 abaixo.

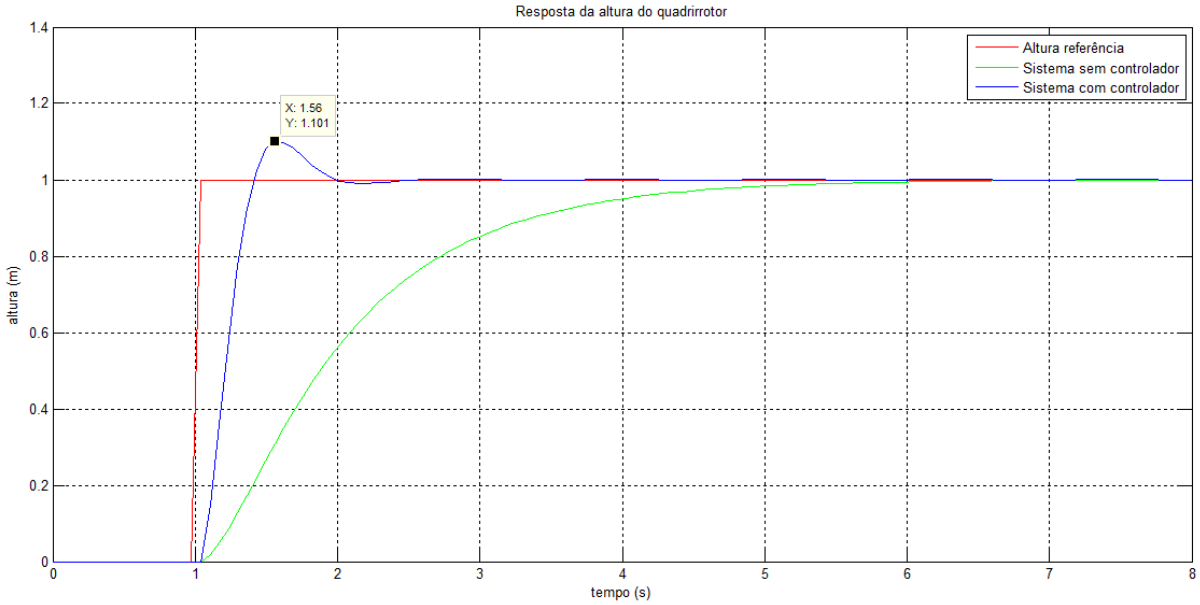

Figura 5. 18 – Resposta da altura do quadrirrotor.

A resposta do sistema compensado foi de longe mais rápida que o sistema sem compensação, no qual o tempo de assentamento caiu de 3.03 segundos para 0.82 segundo. Neste caso, foi introduzido um sobressinal, que não tinha no sistema sem compensação, com valor de 10.1 % (conforme ilustrado na Figura 5.18) o que está dentro das especificações de projeto.

5.3.2 PROJETO CONTROLADOR COM RESPOSTA *DEADBEAT* PARA A ALTURA

Controladores com resposta *DeadBeat* são característicos em alcançar o valor desejado da saída com o menor tempo possível [3].

O controlador com resposta DeadBeat possui a seguinte forma geral:

$$G_d(\mathrm{z}) = \frac{1}{\mathrm{G}(\mathrm{z})}\frac{M(z)}{1-M(z)}, \tag{5.9}$$

onde $\mathrm{G}(\mathrm{z})$ é a função de transferência do sistema, no caos Z_a, $M(z)$ é a função de transferência em malha fechada do sistema junto ao controlador, conforme ilustra a Figura 5.15 e é definida como:

$$\mathrm{M}(\mathrm{z}) = \frac{G_d(\mathrm{z})\mathrm{G}(\mathrm{z})}{1+G_d(\mathrm{z})\mathrm{G}(\mathrm{z})}. \tag{5.10}$$

Além disso, $\mathrm{M}(\mathrm{z})$ possui os zeros e $1-\mathrm{M}(\mathrm{z})$ os polos do sistema em malha aberta que se encontram sobre ou fora do Círculo de Raio Unitário do LGR. Dessa forma o controlador não possui zeros e polos fora da região de estabilidade do sistema. Assim, o controlador projetado tem a seguinte equação:

$$G_d(z) = 55.88\frac{z - 0.685}{z + 0.01296}. \quad (5.11)$$

A simulação foi então realizada conforme diagrama de blocos mostrado na Figura 5.19.

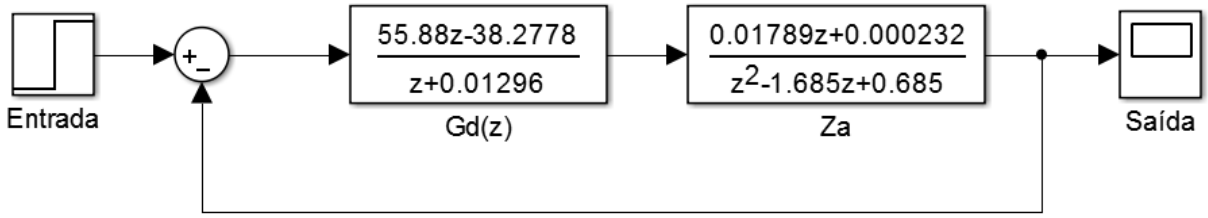

Figura 5. 19 – Diagrama de blocos de simulação do controlador DeadBeat projetado da altura.

A resposta DeadBeat é então apresentada na Figura 5.20.

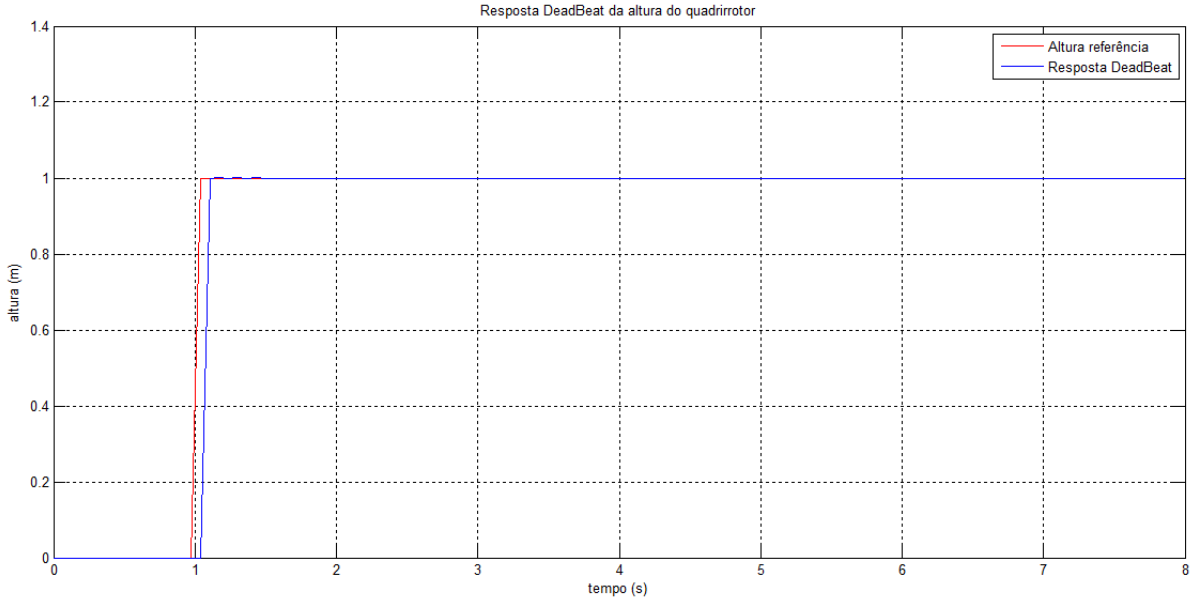

Figura 5. 20 – Resposta DeadBeat da altura.

Conforme o esperado a resposta *DeadBeat* da altura do quadrirrotor alcançou a referência no menor tempo possível, que neste caso, é o tempo de amostragem do sistema 0.065 segundo.

5.3.3 PROJETO CONTROLADOR POR INTERMÉDIO DO LGR PARA ARFAGEM

Assim como foi realizado na seção 5.3.1 para a dinâmica da altura, projetou-se um controlador por intermédio do LGR, dessa vez com outras especificações conforme mostrado abaixo:

$$M_p = 10\ \%\ e\ t_s = 0.5\ s. \quad (5.12)$$

Dessa forma o coeficiente de amortecimento é $\xi = 0.5912$ e a frequência natural de oscilação $w_n = 13.5328$. Os polos em tempo contínuo do projeto, então, são:

$$s = -8.000 \pm j10.915, \tag{5.13}$$

os quais obtêm os polos em tempo discreto, através da equação (5.14):

$$z = 0.4511 \pm j0.3873. \tag{5.14}$$

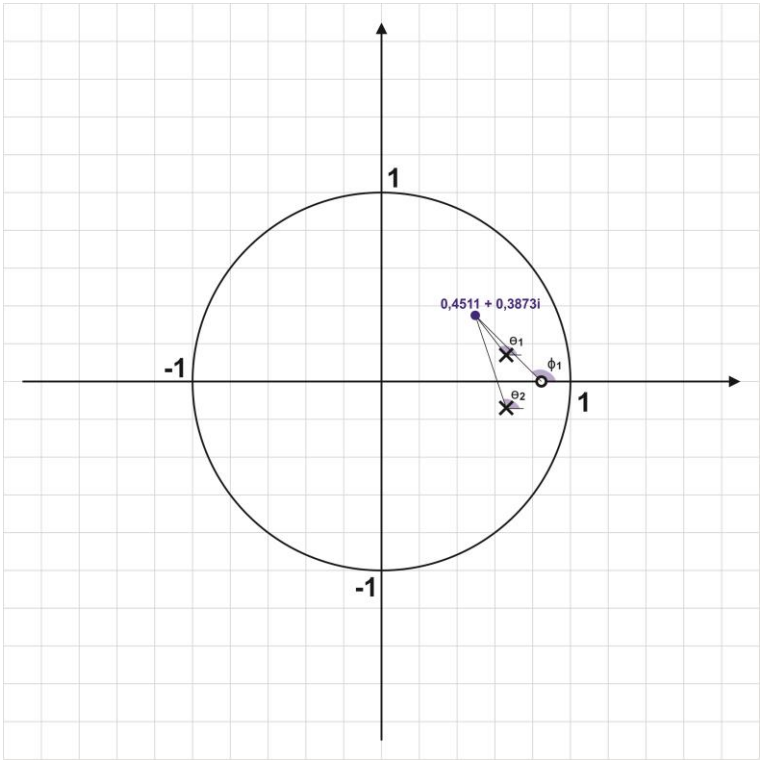

Figura 5. 21 – LGR para cálculo dos ângulos dos polos e zeros da arfagem.

O cálculo do valor do ângulo (Figura 5.21) pelo qual o controlador deve obter é feito utilizando a equação (5.15), resultando:

$$\theta_c = -40.87º\,. \tag{5.15}$$

Devido ao valor negativo do ângulo de compensação do sistema, o controlador a ser projetado é PI – Proporcional Integral, pelo qual sua forma geral é apresentada na equação (5.16).

$$G_d(z) = K\frac{z - z_c}{z - p_c}, onde\ z_c < p_c \tag{5.16}$$

Conforme mostrado na seção 3.5.2 pela Figura 3.10, o sistema não compensado possui erro em regime permanente alto. A fim de zerar este erro é necessário aumentar o tipo do sistema, dessa forma, adicionando um integrador, ou seja:

$$p_c = 1 \tag{5.17}$$

A partir daí projetou-se o zero e o ganho do controlador. Obtivemos então compensador por atraso de fase conforme mostra a equação (5.18).

$$G_d(z) = 6.1731\frac{z - 0.739}{z - 1} \tag{5.18}$$

A simulação foi feita conforme diagrama de blocos da Figura 5.22.

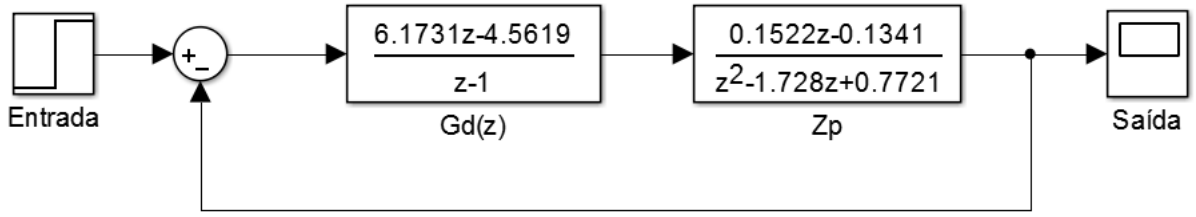

Figura 5. 22 – Diagrama de blocos de simulação do controlador projetado da arfagem.

A resposta para uma entrada degrau com amplitude igual a 0.139 radianos (8º) pode ser visualizada pela Figura 5.23.

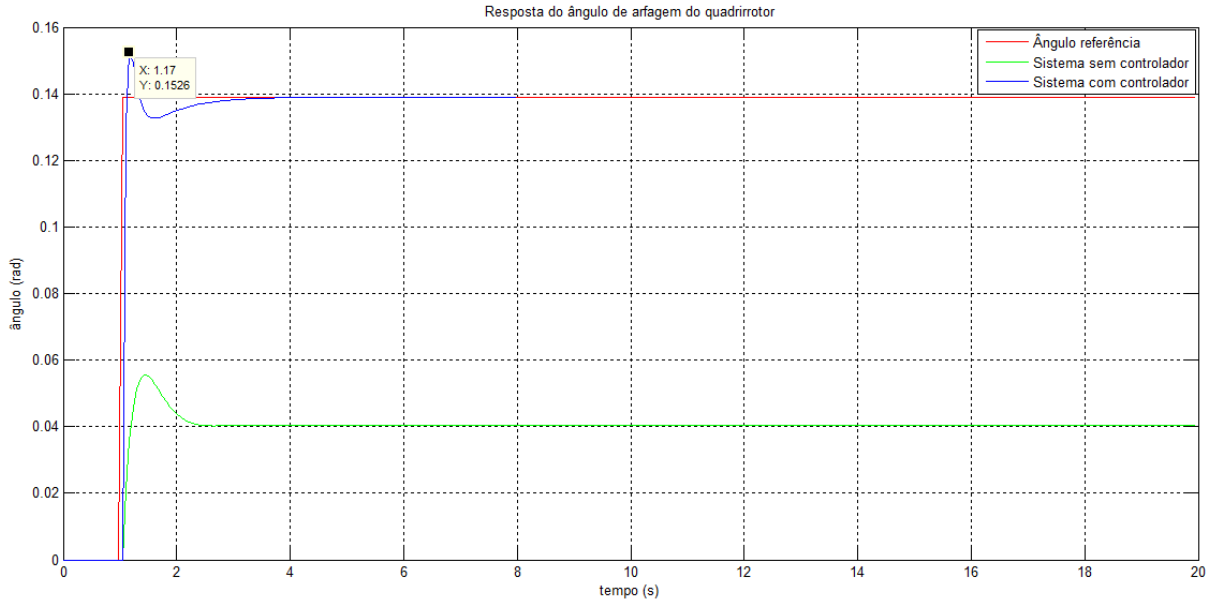

Figura 5. 23 – Resposta do ângulo de arfagem do quadrirrotor.

Como observado na Figura 5.23, o sistema com compensador em cascata melhorou e muito a resposta, conseguiu alcançar erro nulo em regime permanente e o sobressinal obtido ficou dentro das especificações do projeto (9.7 %), além de que o tempo de assentamento foi de 0.3 segundo, abaixo do valor especificado.

5.3.4 PROJETO CONTROLADOR COM RESPOSTA *DEADBEAT* PARA ARFAGEM

Como já mencionado o controlador com resposta *DeadBeat* tende a obter polos e zeros que estejam dentro do Círculo de Raio Unitário garantindo assim estabilidade ao sistema, dessa forma, o controlador projetado para o ângulo de arfagem foi o da equação (5.19).

$$G_d(z) = 6.5695\frac{z^2 - 1.728z + 0.7721}{z^2 - 1.8815z + 0.8815} \tag{5.19}$$

A simulação foi realizada com entrada degrau de amplitude 0.139 radiano (8º) conforme diagrama de blocos da Figura 5.24.

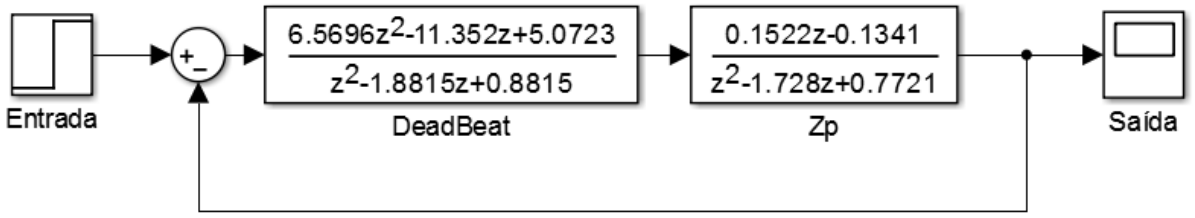

Figura 5. 24 – Diagrama de blocos de simulação do controlador DeadBeat projetado do ângulo de arfagem.

A resposta do sistema é mostrada na Figura 5.25.

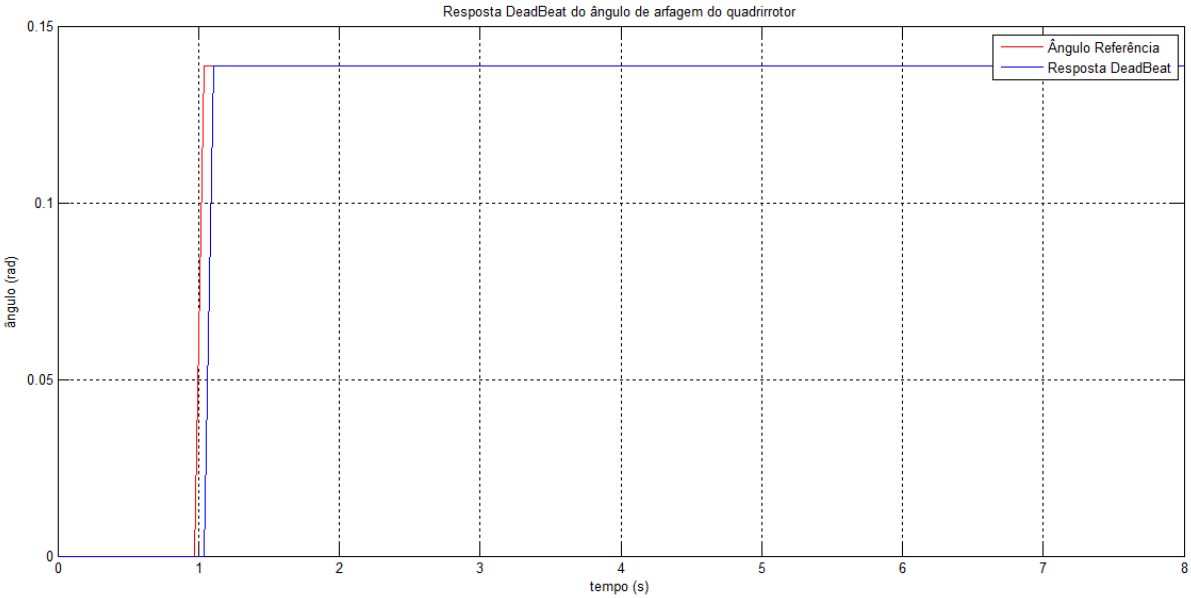

Figura 5. 25 – Resposta DeadBeat do ângulo de arfagem.

Mais uma vez a resposta *DeadBeat* do sistema ocasionou no alcance da referência no menor tempo possível, 0.065 segundo.

5.4 SIMULAÇÃO DE TRAJETÓRIA

A partir dos controladores projetados por intermédio do LGR nos itens anteriores partiu-se para a simulação de uma trajetória pela qual o AR.Drone poderia percorrer. Cada movimento seria realizado separadamente, em tempos distintos.

O kit de desenvolvimento do AR.Drone no Simulink nos fornece a função de transferência que transforma o ângulo de arfagem em velocidade linear do quadrirrotor, denominado $Z_p 2U$, conforme mostrado na equação (5.20).

$$Z_p 2\mathrm{U} = \frac{0.3915}{\mathrm{z} - 0.9577} \qquad (5.20)$$

Com um integrador em cascata no sistema, obtém-se então a posição do quadrirrotor. Além disso, é possível constatar que o sistema de arfagem com compensador em atraso responde em velocidade linear muito mais rápido que o sistema não compensado, podendo ser observado na Figura 5.26.

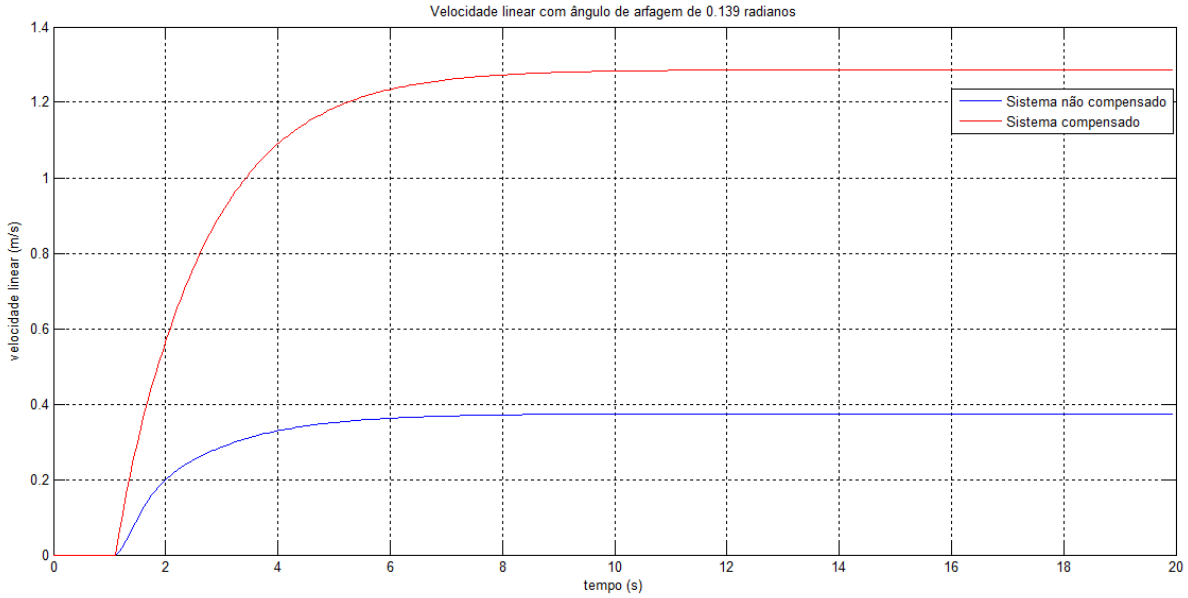

Figura 5. 26 – Velocidade linear do quadrirrotor com ângulo de arfagem de 0.139 radiano.

Realizando a integração da velocidade obtida na Figura 5.26 temos o deslocamento linear realizado pelo quadrirrotor, de modo que obtemos a Figura 5.27.

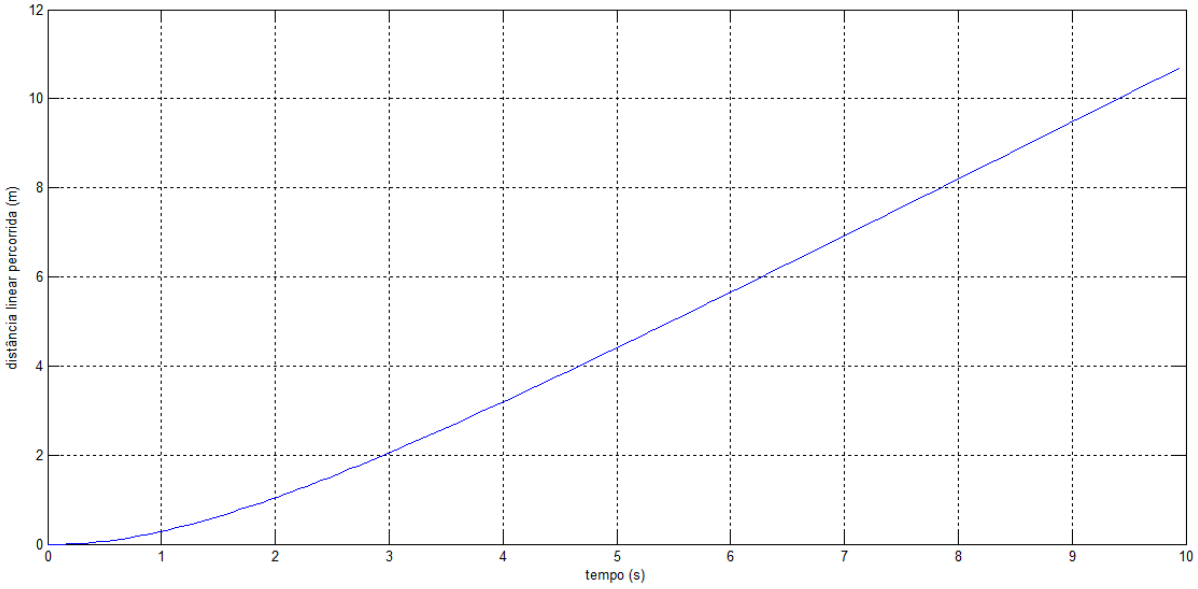

Figura 5. 27 – Deslocamento linear com ângulo de arfagem de 0.139 radiano.

Para obter a função tempo pelo deslocamento linear foi invertido o gráfico e calculado por interpolação, o que nos fornece o tempo pelo qual o quadrirrotor deverá permanecer no movimento de *pitch* para percorrer uma determinada distância. Esse processo pode ser visto na Figura 5.28.

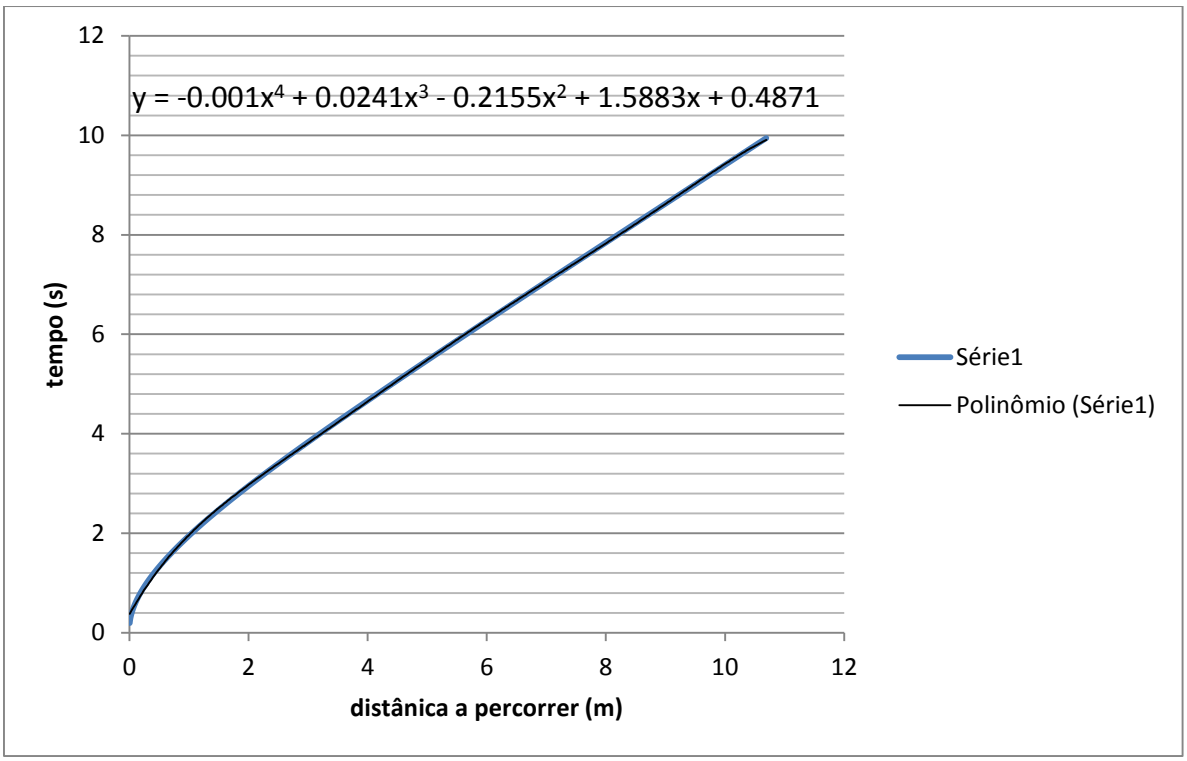

Figura 5. 28 – Relação tempo por distância linear.

Para quantificar a melhoria no desempenho com os controladores desenvolvidos supôs-se uma trajetória no qual o AR.Drone se encontra a 1 metro de altura e identifica um local de pouso a 2 metros em sua direção X (movimento *pitch* – arfagem) e 1.5 metro na direção Y (movimento *roll* – rolamento). Dessa forma o AR.Drone deveria realizar os três movimentos, *pitch*, *roll* e de descida (*throttle*).

O sistema sem controlador realiza essa trajetória em (*pitch* – 5.135 s, *roll* – 4.16 s, Descida – 3.03 s) 12.325 segundos, enquanto que o sistema controlado percorre em (*pitch* – 2.925 s, *roll* – 2.47 s, Descida – 0.95 s) 6.345 segundos, o que podemos concluir que o sistema controlado responde em torno de 50 % mais rápido que o sistema não controlado, evidenciando a importância da utilização de controladores em aplicações de dispositivos autônomos.

Capítulo 6

CONCLUSÃO

6.1 CONCLUSÕES E COMENTÁRIOS FINAIS

Tentou-se controlar o quadrirrotor via comunicação Wi-Fi e porta *udp* pelo kit de desenvolvimento do SIMULINK via MATLAB [20]. Os dados enviados do Ar.Drone para o PC não traduzia o real comportamento do voo, pois indicava que o quadrirrotor estava em uma altura na ordem de quilômetros. Dessa forma, todo o resultado foi obtido via simulação. O modelo do ângulo de arfagem encontrado no kit foi confrontado com outro modelo identificado em [3] o que deu mais segurança na utilização do mesmo.

Sabe-se que o modelo de um quadrirrotor é não-linear e que para ser representado por modelos lineares, primeiramente foi estudado cada movimento separadamente e estabelecidos um modelo para cada. Além disso a ação de controle foi restringida em certos pontos a fim de validar as aproximações realizadas na identificação. Os controladores foram projetados, então, ao redor desses pontos de linearidade e foram validados via simulação. Em simulação todos os controladores se comportaram como o esperado e respeitando as especificações pré-estabelecidas que foram elaboradas conforme limitações físicas do corpo.

A resposta de cada um dos sistemas controlados foi mais rápidoa em relação aos sistemas não controlados, de: 74% para a altura e 73% para o movimento de arfagem.

Os algoritmos elaborados para o processamento de imagens serviram de grande aprendizado. Diversas estruturas foram elaboradas a fim de melhorar os resultados. Constatou-se que algoritmos de extração e descrição de características são dispendiosos computacionalmente e o estudo de melhorias nessa área é muito crescente. A segmentação por cor gerou o melhor dos resultados, porém, com o ambiente controlado. Sabe-se que uma ferramenta prática que funcione em qualquer ambiente deve conter o maior número de redundâncias possível de forma a não ocorrer erros e falhas no sistema.

Mesmo com a baixa resolução da câmera do Ar.Drone os resultados foram significativos, com o algoritmo SURF obteve-se 47.7% de eficiência na identificação dos frames, isso quer dizer que em todos os frames possíveis de serem identificados (com o alvo na imagem), o SURF identificou quase a metade deles. Adicionando o KLT como algoritmo secundário a eficiência aumentou para 66.3% de identificação. Além disso, comparando diretamente o SURF e o KLT, o segundo algoritmo proporcionou melhora de 33.4% em média em relação ao primeiro.

Todos os códigos criados foram executados no software MATLAB versão 2013b em uma máquina com processador i3 2.40 GHz. os tempos de execução para a identificação em um frame apenas, foram: SURF – 0.16 segundo, KLT – 0.14 segundo e segmentação por cor – 0.02 segundo. Esse resultado comprovou que algoritmos de identificação por extração e descrição de características são computacionalmente mais complexos. Para utilizar esses algoritmos em aplicação real seria necessário analisar o frame a cada: 10 frames – SURF; 7 frames – KLT; 2 frames – segmentação por cor.

Apesar de todas as dificuldades encontradas, este trabalho foi concluído com êxito e abordou tópicos da engenharia de grande importância. Os resultados apresentados apesar de realizados via simulação serviram de base para a comprovação das teorias de forma satisfatória.

6.2 TRABALHOS FUTUROS

A fim de criar outros trabalhos científicos nessa área seguem algumas sugestões:

- Implementar o sistema aqui apresentado de forma a validar as simulações realizadas.
- Realizar otimizações nos algoritmos de extração e descrição de características a fim de torná-los menos dispendiosos computacionalmente e possibilitar melhoras na análise das imagens.
- Criar um sistema de controle que realize vários movimentos em conjunto, o que possibilitaria melhorar ainda mais o desempenho da resposta na ação de pousar o quadrirrotor.
- Criar um sistema que utilizasse os sensores inerciais para estimação de posição e através de coordenadas enviadas ao quadrirrotor realiza-se o controle de pouso.
- Validar as simulações aqui apresentadas de forma prática, embarcando os algoritmos na plataforma do AR.Drone.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] **GONZALEZ, R. C.; WOODS, R. E**. *Processamento de Imagens Digitais*. São Paulo: Edgard Blucher, 2000. 509 p.

[2] **PARROT**. AR.Drone Developer Guide. [S.l.], Fevereiro 2011. SDK 1.7. Disponível em https://projects.ardrone.org/

[3] **LIMA, F.** *Implementação de controle de rolagem e arfagem com realimentação visual aplicado a um quadrirrotor comercial*. Trabalho de Graduação em Engenharia de Controle e Automação, Universidade de Brasília, 2013.

[4] **BAKER, S.; MATTHEWS, I.** *Lucas-Kanade 20 years on: A unifying framework*. International Journal of Computer Vision, 56(3):221 – 255, 2004.

[5] **LUCAS, B. D.; KANADE, T.** *An iteractive image registration technique with an application to stereo vision.* In Proceedings of the 7th International Conference on Artificial Intelligence, páginas 674 – 679, August 1981.

[6] **HARRIS, C.; STEPHEN, M.** *A combined corner and edge detection*. In M. M. Matthews editor. Proceedings of the 4th ALVEY vision conference, páginas 147 – 151, University of Manchester, England, Setembro 1988.

[7] **SHI, J.; TOMASI, C.** *Good features to track.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), páginas 593 – 600, 1994.

[8] **TOMASI, C.; KANADE, T**. *Detection and tracking of point features*. Technical Report CMU – CS – 91 – 132, Carnegie Mellon University, 1991.

[9] **PARKER, J. R.** In:___. *Algorithms for image processing and computer vision.* New York: John Wiley & Sons, c1997. 417 p.

[10] **SZELISKI, R.** In:___. *Computer vision: algorithms and applications.* London: Springer, c2011. 812p.

[11] **FORSYTH, D.; PONCE, J.** *Computer vision: a modern approach.* 2nd ed. Boston, Columbus: Pearson, 2012. 761 p.

[12] **BROWN, M; LOWE, D**. *Invariant features from interest point groups*. In: BMVC. (2002).

[13] **AFONSO, J, M, P.** *Framework para sistemas de navegação de veículos aéreos não tripulados*. Monografia (Bacharelado em Engenharia da Computação), Universidade Federal de Ouro Preto, 2014.

[14] **MARTIN, G.** *Modelling and Control of the Parrot AR.Drone.* Final Project Report UNSW Canberra, 2012.

[15] **BRESCIANI, T.** *Modelling, Identification and Control of a Quadrotor Helicopter*. Tese (Mestrado) - Lund University, 2008.

[16] **KRAJNÌK,T.; VONASEK, V.; FISER, D; FAIGL, J**. *AR-Drone as a Platform for Robotic Research and Education*. In: Research and Education in Robotics: 2011, Heidelberg, Springer, 2011.

[17] **HANSEN, J. P.; ALAPETITE, A.; MACKENZIE, I. S.; MOLLENBACH, E**. *The use of gaze to control drones*. Proceedings of the ACM Symposium on Eye Tracking Research and Applications – ETRA. New York, 2014.

[18] **NISE, N, S**. *Engenharia de Sistemas de Controle*. LTC, 6ª edição. Rio de Janeiro, 2012.

[19] **PEDRINI, H.; SCHWARTZ, W. R**. *Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações.* São Paulo: Thomson Learning, 2008.

[20] **MATHWORKS**. *AR Drone Simulink Development-Kit V1.1*. Estados Unidos, 01 de out. 2013. Disponível em http://www.mathworks.com/matlabcentral/fileexchange/43719-ar-drone-simulink-development-kit-v1-> Acesso em: 10 de junho de 2015.